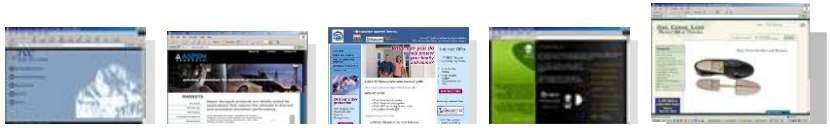


Review

Grant COMMUNICATIONS LLC

Site Assessment™

Latent Semantic Indexing

Submitted: February 24, 2014

Get Results

A more technical document for those folks who need to go to sleep early this evening…

Table of Contents

“I keep hearing LSI being mentioned. What is it?”

Good question! “LSI” stands for “Latent Semantic Indexing”. We have the snapshot of Wikipedia below, as well as our in-depth review of the statistical operations Google is conducting using cloud computing on a truly massive scale. As many as 48 trillion pages being filtered by this 3D approach to quantifying ranking placement.

This is an incomplete work that is evolving as we penetrate the Googleplex. I’m not a mathematician, but I DO understand logic and data sets. What follows is a logical extension of what we’ve uncovered through 1/15/13; approximately 400 man hours invested since 10/25/12 when we had enough data to understand that the last Panda update on 9/28/12 from Google was structurally monumental. And hugely different.

(Note: Every other forum, Google’s Matt Cutts; everyone else, *KNOWS* the update took place 10/5/12. **WRONG ANSWER**!).

One key criteria of the new algorithm is text, content, quality and quantity that leads to value and authority. So why did I spend some free time writing this and doing research? Because I’m up to 950 words (5,729 characters, no white space; 6,697 characters with white space, 59 unique paragraphs), uniquely written, on 1/18/13. I’ll get credit for it and will update my “Author Credibility’. Get ready for the new reality and the reason why this is important.

Really…?

What Google is doing, and how they're doing it?

Excerpt: My friend's name is Jane Doe, which is quite a common name. One day she decided to look it up on Google and, to her utmost surprise, discovered she totally dominated the SERPS - the first page of search results consisted exclusively of her Facebook, LinkedIn, MySpace and other accounts.

Very excited about the finding, she hurried to share it with her friends, including me. However, I told her it was only because Google must have personalized her search results. I said: "Google has become so smart nowadays it even knows what particular Jane Doe you are looking for".

Today Google, Bing and other search engines bias their search results to tailor them to a particular user. This means two searchers looking for one and the same term might see quite different listings.

To collect information about users, Google and Bing keep track of:

- previous queries (for instance, if you look for "hotels" and right after that search for "Nevada", Google might include results for "hotels Nevada")
- web history

- geographic position (they identify your IP address)
- browser language
- social media connections (Google recently introduced Google Social Search that incorporates search results from one's friends and contacts on Google accounts, Twitter or Facebook. Bing, in its turn, has been integrated with Facebook for quite a while now).

Now, what and how are they doing this?

What:

Google has key problems in the areas of parallel processing, data mining, bioinformatics, and collaborative filtering.

The problem: clustering low- and high-dimensional datasets and for analyzing the characteristics of the various clusters. Google must be agile enough for clustering data sets arising in many diverse application areas including information retrieval, customer purchasing transactions, web, GIS, science, biology and general business by every vertical market. Independently, in order to maintain a relative logarithmic scale over rank and sort criteria.

Relative vs. Absolute by Users:

Additionally, they would need to convert from a relative search phrase to an absolute Boolean logic operator, without the user needing to know Boolean logic.

For instance, a relative search phrase would be "MA Web Design" equals "Web Design MA"; this search includes these three words "MA", "Web", "Design". Any order would produce the same, identical ranking results.

Now, enter this as an absolute search parameter; "MA Web Design". This is going to be ranked differently that "Web Design MA", and is all based in context of the 200+ competitive data sets and corresponding Latent Semantic Indexing within the markets Google has identified as being silo users.

Google & Cloud Computing

Why was Google was core to the introduction of Cloud Computing? They needed the raw horsepower. To conduct our admittedly tiny replication of Google in 2D format, we used a core 6-processor system with 32Gb RAM and 2Gb dedicated VRAM. Even with this computing resource, it still takes two hours to open the core files to analyze a domain and measured page on the 450 parameters and elements we've identified to date. What Google needs is far beyond that for correct interpretation and ranking conflict resolution.

Parameters Needed:

- Multiple classes of clustering algorithms:
  - partitional, agglomerative, & graph-partitioning based.
- Multiple similarity/distance functions:
  - Euclidean distance, cosine, correlation coefficient, extended Jaccard
- clustering criterion functions and agglomerative merging.
- agglomerative merging schemes:
  - single-link, complete-link, UPGMA
- cluster visualization:
  - postscript, SVG, gif, xfig, etc.
- summarizing the clusters:
  - dimensions, cliques, and frequent itemsets.
- scale to datasets containing billions of objects and hundreds of thousands of dimensions.

How:

The following has been collected and condensed from a series of sources that discuss relational relativity in an absolute fluid data environment. This is done in conjunction with the knowledge that Google has employed LSI to site above text content silos in determining both relevancy and authority for each site/domain as part of an overall statistical universe.

So, what is Facebook's newest addition to search; what exactly do they call it?

Facebook has released a new semantic search engine called Graph Search. So what does that mean for Facebook users? Source

What Google would need:

Partitioning graphs, partitioning finite element meshes, and producing fill reducing orderings for sparse matrices. The algorithms implemented probably are based on multilevel recursive-bisection, multilevel *k*-way, and multi-constraint partitioning schemes.

Implement a variety of algorithms for partitioning unstructured graphs, meshes, and for computing fill-reducing orderings of sparse matrices. suited for parallel AMR computations and large scale numerical simulations. The algorithms are based on the parallel multilevel *k*-way graph-partitioning, adaptive repartitioning, and parallel multi-constrained partitioning schemes

Partitioning hypergraphs such as those corresponding to VLSI circuits. The algorithms implemented are based on the multilevel hypergraph partitioning schemes

Graphical application for clustering low- and high-dimensional datasets and for analyzing the characteristics of the various clusters.

Web-enabled data clustering application that is designed for the clustering and data-analysis requirements of gene-expression analysis. Network Visualization Users select from a number of clustering methods, perform the analysis on the server, and visualize the final rank results.

Core Data Sets for Visualization:

Multiple classes of clustering algorithms:

- Partitional, agglomerative, & graph-partitioning based.
- Multiple similarity/distance functions:
  - Euclidean distance, cosine, correlation coefficient, extended Jaccard, user-defined.
- Numerous novel clustering criterion functions and agglomerative merging schemes.
- Traditional agglomerative merging schemes:
  - single-link, complete-link, UPGMA
- Extensive cluster visualization capabilities and output options:
  - postscript, SVG, gif, xfig, etc.
- Multiple methods for effectively summarizing the clusters:
  - most descriptive and discriminating dimensions, cliques, and frequent itemsets.
- Can scale to very large datasets containing hundreds of thousands of objects and tens of thousands of dimensions.

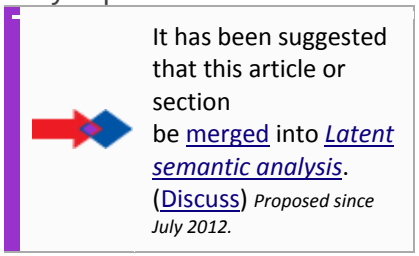
Now, what is the philosophy behind Googles' benchmarking?

Latent Semantic Indexing

Contents

[hide]

From Wikipedia, the free encyclopedia



Latent semantic indexing (LSI) is an indexing and retrieval method that uses a mathematical technique called Singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text. LSI is based on the principle that words that are used in the same contexts tend to have similar meanings. A key feature of LSI is its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts.[1]

LSI is also an application of correspondence analysis, a multivariate statistical technique developed by Jean-Paul Benzécri[2] in the early 1970s, to a contingency table built from word counts in documents.

Called Latent Semantic Indexing because of its ability to correlate semantically related terms that are latent in a collection of text, it was first applied to text at Bell Laboratories in the late 1980s. The method, also called latent semantic analysis (LSA), uncovers the underlying latent semantic structure in the usage of words in a body of text and how it can be used to extract the meaning of the text in response to user queries, commonly referred to as concept searches. Queries, or concept searches, against a set of documents that have undergone LSI will return results that are conceptually similar in meaning to the search criteria even if the results don't share a specific word or words with the search criteria.

[edit]Benefits of LSI

LSI overcomes two of the most problematic constraints of Boolean keyword queries: multiple words that have similar meanings (synonymy) and words that have more than one meaning (polysemy). Synonymy is often the cause of mismatches in the vocabulary used by the authors of documents and the users of information retrieval systems.[3][4] As a result, Boolean or keyword queries often return irrelevant results and miss information that is relevant.

LSI is also used to perform automated document categorization. In fact, several experiments have demonstrated that there are a number of correlations between the way LSI and humans process and categorize text.[5] Document categorization is the assignment of documents to one or more predefined categories based on their similarity to the conceptual content of the categories.[6] LSI uses *example* documents to establish the conceptual basis for each category. During categorization processing, the concepts contained in the documents being categorized are compared to the concepts contained in the example items, and a category (or categories) is assigned to the documents based on the similarities between the concepts they contain and the concepts that are contained in the example documents.

Dynamic clustering based on the conceptual content of documents can also be accomplished using LSI. Clustering is a way to group documents based on their conceptual similarity to each other without using example documents to establish the conceptual basis for each cluster. This is very useful when dealing with an unknown collection of unstructured text.

Because it uses a strictly mathematical approach, LSI is inherently independent of language. This enables LSI to elicit the semantic content of information written in any language without requiring the use of auxiliary structures, such as dictionaries and thesauri. LSI can also perform cross-linguistic concept searching and example-based categorization. For example, queries can be made in one language, such as English, and conceptually similar results will be returned even if they are composed of an entirely different language or of multiple languages.

LSI is not restricted to working only with words. It can also process arbitrary character strings. Any object that can be expressed as text can be represented in an LSI vector space.[7] For example, tests with MEDLINE abstracts have shown that LSI is able to effectively classify genes based on conceptual modeling of the biological information contained in the titles and abstracts of the MEDLINE citations.[8]

LSI automatically adapts to new and changing terminology, and has been shown to be very tolerant of noise (i.e., misspelled words, typographical errors, unreadable characters, etc.).[9] This is especially important for applications using text derived from Optical Character Recognition (OCR) and speech-to-text conversion. LSI also deals effectively with sparse, ambiguous, and contradictory data.

Text does not need to be in sentence form for LSI to be effective. It can work with lists, free-form notes, email, Web-based content, etc. As long as a collection of text contains multiple terms, LSI can be used to identify patterns in the relationships between the important terms and concepts contained in the text.

LSI has proven to be a useful solution to a number of conceptual matching problems.[10][11] The technique has been shown to capture key relationship information, including causal, goal-oriented, and taxonomic information.[12]

[edit]LSI Timeline

Mid-1960s – Factor analysis technique first described and tested (H. Borko and M. Bernick)
1988 – Seminal paper on LSI technique published (Deerwester et al.)
1989 – Original patent granted (Deerwester et al.)
1992 – First use of LSI to assign articles to reviewers[13] (Dumais and Nielsen)
1994 – Patent granted for the cross-lingual application of LSI (Landauer et al.)
1995 – First use of LSI for grading essays (Foltz, et al., Landauer et al.)
1999 – First implementation of LSI technology for intelligence community for analyzing unstructured text (SAIC).
2002 – LSI-based product offering to intelligence-based government agencies (SAIC)
2005 – First vertical-specific application – publishing – EDB (EBSCO, Content Analyst Company)

[edit]Mathematics of LSI

LSI uses common linear algebra techniques to learn the conceptual correlations in a collection of text. In general, the process involves constructing a weighted term-document matrix, performing a **Singular Value Decomposition** on the matrix, and using the matrix to identify the concepts contained in the text.

[edit]**Term Document Matrix**

LSI begins by constructing a term-document matrix, A, to identify the occurrences of the m unique terms within a collection of n documents. In a term-document matrix, each term is represented by a row, and each document is represented by a column, with each matrix cell, a_{ij}, initially representing the number of times the associated term appears in the indicated document, $\mathrm{tf_{ij}}$. This matrix is usually very large and very sparse.

Once a term-document matrix is constructed, local and global weighting functions can be applied to it to condition the data. The weighting functions transform each cell, a_{ij} of A, to be the product of a local term weight, l_{ij}, which describes the relative frequency of a term in a document, and a global weight, g_i, which describes the relative frequency of the term within the entire collection of documents.

Some common local weighting functions [14] are defined in the following table.

Binary	$l_{ij} = 1$ if the term exists in the document, or else 0
TermFrequency	$l_{ij} = \mathrm{tf}_{ij}$, the number of occurrences of term i in document j
Log	$l_{ij} = \log(\mathrm{tf}_{ij} + 1)$
Augnorm	$l_{ij} = \dfrac{\left(\frac{\mathrm{tf}_{ij}}{\max_i(\mathrm{tf}_{ij})}\right) + 1}{2}$

Some common global weighting functions are defined in the following table.

Binary		$g_i = 1$
Normal		$g_i = \dfrac{1}{\sqrt{\sum_j \mathrm{tf}_{ij}^2}}$
GfIdf		$g_i = \mathrm{gf}_i/\mathrm{df}_i$, where gf_i is the total number of times term i occurs in the whole collection, and df_i is the number of documents in which term i occurs.
Idf		$g_i = \log_2 \dfrac{n}{1 + \mathrm{df}_i}$
Entropy		$g_i = 1 + \sum_j \dfrac{p_{ij} \log p_{ij}}{\log n}$, where $p_{ij} = \dfrac{\mathrm{tf}_{ij}}{\mathrm{gf}_i}$

Empirical studies with LSI report that the Log Entropy weighting functions work well, in practice, with many data sets.[15] In other words, each entry a_{ij} of A is computed as:

$$g_i = 1 + \sum_j \frac{p_{ij} \log p_{ij}}{\log n}$$

$$a_{ij} = g_i \ \log(\mathrm{tf}_{ij} + 1)$$

[edit]Rank-Reduced Singular Value Decomposition

A rank-reduced, Singular Value Decomposition is performed on the matrix to determine patterns in the relationships between the terms and concepts contained in the text. The SVD forms the foundation for LSI.[16] It computes the term and document vector spaces by transforming the single term-frequency matrix, A, into three other matrices— an ***m*** by ***r*** term-concept vector matrix T, an ***r*** by ***r*** singular values matrix S, and a ***n*** by ***r*** concept-document vector matrix, D, which satisfy the following relations:

$$A = TSD^T$$

$$T^T T = I_r \quad D^T D = I_r$$

$$S_{1,1} \geq S_{2,2} \geq \ldots \geq S_{r,r} > 0 \quad S_{i,j} = 0 \text{ where } i \neq j$$

In the formula, **A** is the supplied ***m*** by ***n*** weighted matrix of term frequencies in a collection of text where ***m*** is the number of unique terms, and ***n*** is the number of documents. **T** is a computed ***m*** by ***r*** matrix of term vectors where ***r*** is the rank of **A**—a measure of its unique dimensions **≤ min(*m*,*n*)**. **S** is a computed ***r*** by ***r*** diagonal matrix of decreasing singular values, and **D** is a computed ***n*** by ***r*** matrix of document vectors.

The LSI modification to a standard SVD is to reduce the rank or truncate the singular value matrix **S** to size ***k*** « ***r***, typically on the order of a ***k*** in the range of 100 to 300 dimensions, effectively reducing the term and document vector matrix sizes to ***m*** by ***k*** and ***n*** by ***k*** respectively. The SVD operation, along with this reduction, has the effect of preserving the most important semantic information in the text while reducing noise and other undesirable artifacts of the original space of **A**. This reduced set of matrices is often denoted with a modified formula such as:

$$\mathbf{A} \approx \mathbf{A}_k = \mathbf{T}_k \mathbf{S}_k \mathbf{D}_k^T$$

Efficient LSI algorithms only compute the first ***k*** singular values and term and document vectors as opposed to computing a full SVD and then truncating it.

Note that this rank reduction is essentially the same as doing Principal Component Analysis (PCA) on the matrix **A**, except that PCA subtracts off the means. PCA provides cleaner mathematics, but loses the sparseness of the **A** matrix, which can make it infeasible for large lexicons.

[edit]Querying and Augmenting LSI Vector Spaces

The computed $\mathbf{T}_k$ and $\mathbf{D}_k$ matrices define the term and document vector spaces, which with the computed singular values, $\mathbf{S}_k$, embody the conceptual information derived from the document collection. The similarity of terms or documents within these spaces is a factor of how close they are to each other in these spaces, typically computed as a function of the angle between the corresponding vectors.

The same steps are used to locate the vectors representing the text of queries and new documents within the document space of an existing LSI index. By a simple transformation of the $\mathbf{A = T S D}^T$ equation into the equivalent $\mathbf{D = A}^T \mathbf{T S}^{-1}$ equation, a new vector, $\boldsymbol{d}$, for a query or for a new document can be created by computing a new column in **A** and then multiplying the new column by $\mathbf{T S}^{-1}$. The new column in **A** is computed using the originally derived global term weights and applying the same local weighting function to the terms in the query or in the new document.

A drawback to computing vectors in this way, when adding new searchable documents, is that terms that were not known during the SVD phase for the original index are ignored. These terms will have no impact on the global weights and learned correlations derived from the original collection of text. However, the computed vectors for the new text are still very relevant for similarity comparisons with all other document vectors.

The process of augmenting the document vector spaces for an LSI index with new documents in this manner is called *folding in*. Although the folding-in process does not account for the new semantic content of the new text, adding a substantial number of documents in this way will still provide good results for queries as long as the terms and concepts they contain are well represented within the LSI index to which they are being added. When the terms and concepts of a new set of documents need to be included in an LSI index, either the term-document matrix, and the SVD, must be recomputed or an incremental update method (such as the one described in [17]) be used.

[edit]Additional Uses of LSI

It is generally acknowledged that the ability to work with text on a semantic basis is essential to modern information retrieval systems. As a result, the use of LSI has significantly expanded in recent years as earlier challenges in scalability and performance have been overcome.

LSI is being used in a variety of information retrieval and text processing applications, although its primary application has been for concept searching and automated document categorization.[18] Below are some other ways in which LSI is being used:

- Information discovery[19] (eDiscovery, Government/Intelligence community, Publishing)
- Automated document classification (eDiscovery, Government/Intelligence community, Publishing)[20]
- Text summarization[21] (eDiscovery, Publishing)

- Relationship discovery[22] (Government, Intelligence community, Social Networking)
- Automatic generation of link charts of individuals and organizations[23] (Government, Intelligence community)
- Matching technical papers and grants with reviewers[24] (Government)
- Online customer support[25] (Customer Management)
- Determining document authorship[26] (Education)
- Automatic keyword annotation of images[27]
- Understanding software source code[28] (Software Engineering)
- Filtering spam[29] (System Administration)
- Information visualization[30]
- Essay scoring[31] (Education)
- Literature-based discovery[32]

LSI is increasingly being used for electronic document discovery (eDiscovery) to help enterprises prepare for litigation. In eDiscovery, the ability to cluster, categorize, and search large collections of unstructured text on a conceptual basis is essential. Concept-based searching using LSI has been applied to the eDiscovery process by leading providers as early as 2003.[33]

[edit]Challenges to LSI

Early challenges to LSI focused on scalability and performance. LSI requires relatively high computational performance and memory in comparison to other information retrieval techniques.[34] However, with the implementation of modern high-speed processors and the availability of inexpensive memory, these considerations have been largely overcome. Real-world applications involving more than 30 million documents that were fully processed through the matrix and SVD computations are not uncommon in some LSI applications.

Another challenge to LSI has been the alleged difficulty in determining the optimal number of dimensions to use for performing the SVD. As a general rule, fewer dimensions allow for broader comparisons of the concepts contained in a collection of text, while a higher number of dimensions enable more specific (or more relevant) comparisons of concepts. The actual number of dimensions that can be used is limited by the number of documents in the collection. Research has demonstrated that around 300 dimensions will usually provide the best results with moderate-sized document collections (hundreds of thousands of documents) and perhaps 400 dimensions for larger document collections (millions of documents).[35] However, recent studies indicate that 50-1000 dimensions are suitable depending on the size and nature of the document collection.[36]

Checking the amount of variance in the data after computing the SVD can be used to determine the optimal number of dimensions to retain. The variance contained in the data can be viewed by plotting the singular values (S) in a scree plot. Some LSI practitioners select the dimensionality associated with the knee of the curve as the cut-off point for the number of dimensions to retain. Others argue that some quantity of the variance must be retained, and the amount of variance in the data should dictate the proper dimensionality to retain. Seventy percent is often mentioned as the amount of variance in the data that should be used to select the optimal dimensionality for recomputing the SVD.[37][38][39]

[edit]See also

- Latent semantic analysis
- Latent Semantic Structure Indexing
- Principal component analysis
- Correspondence analysis

[edit]References

1. ^ Deerwester, S., et al, Improving Information Retrieval with Latent Semantic Indexing, Proceedings of the 51st Annual Meeting of the American Society for Information Science 25, 1988, pp. 36–40.
2. ^ Benzécri, J.-P. (1973). *L'Analyse des Données. Volume II. L'Analyse des Correspondences*. Paris, France: Dunod.
3. ^ Furnas, G., et al, The Vocabulary Problem in Human-System Communication, Communications of the ACM, 1987, 30(11), pp. 964-971.
4. ^ Zhao, L. and Callan, J., Term Necessity Prediction, Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010). Toronto, Canada, 2010.
5. ^ Landauer, T., et al., Learning Human-like Knowledge by Singular Value Decomposition: A Progress Report, M. I. Jordan, M. J. Kearns & S. A. Solla (Eds.), Advances in Neural Information Processing Systems 10, Cambridge: MIT Press, 1998, pp. 45–51.
6. ^ Dumais, S., Platt J., Heckerman D., and Sahami M., Inductive Learning Algorithms and Representations For Text Categorization, Proceedings of ACM-CIKM'98, 1998.
7. ^ Zukas, Anthony, Price, Robert J., Document Categorization Using Latent Semantic Indexing, White Paper, Content Analyst Company, LLC
8. ^ Homayouni, Ramin, Heinrich, Kevin, Wei, Lai, Berry, Michael W., Gene Clustering by Latent Semantic Indexing of MEDLINE Abstracts, August 2004, pp. 104–115.

9. ^ Price, R., and Zukas, A., Application of Latent Semantic Indexing to Processing of Noisy Text, Intelligence and Security Informatics, Lecture Notes in Computer Science, Volume 3495, Springer Publishing, 2005, pp. 602–603.
10. ^ Ding, C., A Similarity-based Probability Model for Latent Semantic Indexing, Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp. 59–65.
11. ^ Bartell, B., Cottrell, G., and Belew, R., Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling, Proceedings, ACM SIGIR Conference on Research and Development in Information Retrieval, 1992, pp. 161–167.
12. ^ Graesser, A., and Karnavat, A., Latent Semantic Analysis Captures Causal, Goal-oriented, and Taxonomic Structures, Proceedings of CogSci 2000, pp. 184–189.
13. ^ Dumais, S., and Nielsen, J., Automating the Assignment of Submitted Manuscripts to Reviewers, Proceedings of the Fifteenth Annual International Conference on Research and Development in Information Retrieval, 1992, pp. 233–244.
14. ^ Berry, M. W., and Browne, M., Understanding Search Engines: Mathematical Modeling and Text Retrieval, Society for Industrial and Applied Mathematics, Philadelphia, (2005).
15. ^ Landauer, T., et al., Handbook of Latent Semantic Analysis, Lawrence Erlbaum Associates, 2007.
16. ^ Berry, Michael W., Dumais, Susan T., O'Brien, Gavin W., Using Linear Algebra for Intelligent Information Retrieval, December 1994, SIAM Review 37:4 (1995), pp. 573–595.
17. ^ Matthew Brand (2006). "Fast Low-Rank Modifications of the Thin Singular Value Decomposition" (PDF). *Linear Algebra and Its Applications* **415**: 20–30. doi:10.1016/j.laa.2005.07.021.
18. ^ Dumais, S., Latent Semantic Analysis, ARIST Review of Information Science and Technology, vol. 38, 2004, Chapter 4.
19. ^ Best Practices Commentary on the Use of Search and Information Retrieval Methods in E-Discovery, the Sedona Conference, 2007, pp. 189–223.
20. ^ Foltz, P. W. and Dumais, S. T. Personalized Information Delivery: An analysis of information filtering methods, Communications of the ACM, 1992, 34(12), 51-60.
21. ^ Gong, Y., and Liu, X., Creating Generic Text Summaries, Proceedings, Sixth International Conference on Document Analysis and Recognition, 2001, pp. 903–907.
22. ^ Bradford, R., Efficient Discovery of New Information in Large Text Databases, Proceedings, IEEE International Conference on Intelligence and Security Informatics, Atlanta, Georgia, LNCS Vol. 3495, Springer, 2005, pp. 374–380.
23. ^ Bradford, R., Application of Latent Semantic Indexing in Generating Graphs of Terrorist Networks, in: Proceedings, IEEE International Conference on Intelligence and Security Informatics, ISI 2006, San Diego, CA, USA, May 23–24, 2006, Springer, LNCS vol. 3975, pp. 674–675.

24. ^ Yarowsky, D., and Florian, R., Taking the Load off the Conference Chairs: Towards a Digital Paper-routing Assistant, Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in NLP and Very-Large Corpora, 1999, pp. 220–230.
25. ^ Caron, J., Applying LSA to Online Customer Support: A Trial Study, Unpublished Master's Thesis, May 2000.
26. ^ Soboroff, I., et al, Visualizing Document Authorship Using N-grams and Latent Semantic Indexing, Workshop on New Paradigms in Information Visualization and Manipulation, 1997, pp. 43–48.
27. ^ Monay, F., and Gatica-Perez, D., On Image Auto-annotation with Latent Space Models, Proceedings of the 11th ACM international conference on Multimedia, Berkeley, CA, 2003, pp. 275–278.
28. ^ Maletic, J., and Marcus, A., Using Latent Semantic Analysis to Identify Similarities in Source Code to Support Program Understanding, Proceedings of 12th IEEE International Conference on Tools with Artificial Intelligence, Vancouver, British Columbia, November 13–15, 2000, pp. 46–53.
29. ^ Gee, K., Using Latent Semantic Indexing to Filter Spam, in: Proceedings, 2003 ACM Symposium on Applied Computing, Melbourne, Florida, pp. 460–464.
30. ^ Landauer, T., Laham, D., and Derr, M., From Paragraph to Graph: Latent Semantic Analysis for Information Visualization, Proceedings of the National Academy of Science, 101, 2004, pp. 5214–5219.
31. ^ Foltz, Peter W., Laham, Darrell, and Landauer, Thomas K., Automated Essay Scoring: Applications to Educational Technology, Proceedings of EdMedia, 1999.
32. ^ Gordon, M., and Dumais, S., Using Latent Semantic Indexing for Literature Based Discovery, Journal of the American Society for Information Science, 49(8), 1998, pp. 674–685.
33. ^ There Has to be a Better Way to Search, 2008, White Paper, Fios, Inc.
34. ^ Karypis, G., Han, E., Fast Supervised Dimensionality Reduction Algorithm with Applications to Document Categorization and Retrieval, Proceedings of CIKM-00, 9th ACM Conference on Information and Knowledge Management.
35. ^ Bradford, R., An Empirical Study of Required Dimensionality for Large-scale Latent Semantic Indexing Applications, Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, California, USA, 2008, pp. 153–162.
36. ^ Landauer, Thomas K., and Dumais, Susan T., Latent Semantic Analysis, Scholarpedia, 3(11):4356, 2008.
37. ^ Cangelosi, R., Goriely A., Component Retention In Principal Component Analysis With Application to Cdna Microarray Data, BMC Biology Direct 2(2) (2007).
38. ^ Jolliffe, L. T., Principal Component Analysis, Springer-Verlag, New York, (1986).
39. ^ Hu, X., Z. Cai, et al., LSA: First Dimension and Dimensional Weighting, 25th Annual Meeting of the Cognitive Science Society, Boston, MA.

[edit]External links

- Michael Berry's site
- Gensim contains a Python+NumPy implementation of LSI for matrices larger than the available RAM.
- Text to Matrix Generator (TMG) MATLAB toolbox that can be used for various tasks in text mining (TM) specifically i) indexing, ii) retrieval, iii) dimensionality reduction, iv) clustering, v) classification. Most of TMG is written in MATLAB and parts in Perl. It contains implementations of LSI, clustered LSI, NMF and other methods.
- Stanford University Andrew Ng Video on LSI

[edit]Further reading

Berry, M. W., Browne M., Understanding Search Engines: Mathematical Modeling and Text Retrieval, Philadelphia, Society for Industrial and Applied Mathematics, (2005).

Berry, M. W., (Editor), Survey of Text Mining: Clustering, Classification, and Retrieval, New York, Springer, (2004).

Landauer, T., et al., Handbook of Latent Semantic Analysis, Lawrence Erlbaum Associates, 2007.

Manning, C. D., Schutze H., Foundations of Statistical Natural Language Processing, Cambridge, MA, The MIT Press, (1999).