Lessons Learned from 20 Years of Implementing LSI Applications

Roger Bradford [1]

Maxim Analytics, Great Falls, VA 22066 USA

Abstract

This paper summarizes lessons learned, over a period of 20 years, from implementing information systems employing the technique of latent semantic indexing (LSI). The data presented is drawn from 63 projects undertaken over the period 1999 through 2019. Over that period the projects increased in scale from collections of hundreds of thousands of documents to ones involving hundreds of millions of documents. They also increased in sophistication, from simple search and retrieval systems to ones focused on information discovery and automated alerting. This paper summarizes some of the key developments in technology and techniques that enabled those advances in the size and sophistication of the applications. The objective of this paper is to share insights gained from these past two decades of system implementation experience.

Keywords

Latent Semantic Indexing, LSI, LSI applications, LSA, lessons learned

1. Latent Semantic Indexing

The technique of latent semantic indexing (LSI) was invented at Bellcore in the late 1980s [1]. The original intent was to provide improved capabilities for retrieval of text. The technique has, however, proven to be useful in analysis of a wide variety of information types [2, 3].

As applied to a collection of documents, the LSI algorithm consists of the following primary steps [1, 4]:

1. A term-document matrix is formed, and (typically) local and global weights are applied to the elements of this matrix.
2. Singular value decomposition (SVD) is used to reduce this matrix to a product of three matrices, one of which is diagonal in the singular values of the original matrix.
3. Dimensionality is reduced by deleting all but the k largest singular values, together with the corresponding columns of the other two matrices.
4. This truncation process provides a basis for generating a k-dimensional vector space. Both terms and documents are represented by k-dimensional vectors in this vector space.
5. New queries, terms, and documents can be represented in the space by a process known as folding-in, which extrapolates from known vectors.
6. The semantic similarity of any two objects represented in the space is reflected by the proximity of their representation vectors, generally using a cosine measure.

Experience from a broad range of academic, industrial, and governmental testing has shown that proximity in an LSI space is a remarkably good proxy for semantic relatedness as judged by humans [5].

Early commercial applications of LSI included identification of people with specific expertise [6], detection of spam in e-mails [3] and essay scoring [7]. Over time, the technique found wide application in areas such as patent search and analysis [8], résumé matching [9], customer survey analysis [10], and fraud detection [11]. It became the dominant paradigm in electronic

DESIRES 2021 - 2nd International Conference on Design of Experimental Search & Information REtrieval Systems, September 15--18, 2021, Padua, Italy
EMAIL: rbradford@cox.net
ORCID: 0000-0003-1750-3125

document discovery [12]. More recently it has been used in bioinformatics discovery [13], recommender systems [14], and social media analysis [15].

2. Structure of the Paper

This paper summarizes lessons learned from 63 information system implementation projects that the author took part in over the period 1999 through 2019. Each of these projects employed LSI as a key technical component. The systems addressed a wide range of applications for both commercial and government customers.

Over this 20-year period, the systems increased significantly in both size and sophistication.[2] The earliest systems utilized the conceptual search, clustering, and categorization functionality of LSI to implement relatively simple capabilities for such tasks as customer survey analysis and matching of résumés with job openings. Extrapolating from experience gained, these fundamental capabilities subsequently were applied in a more abstract fashion to higher-level considerations in applications such as fraud detection and patent prior art analysis. Successive refinement of tools and techniques eventually enabled advanced applications incorporating features such as novel information detection and secure information sharing.

Section 3 of this paper provides a brief overview of the principal improvements in technologies and techniques that enabled solution of progressively larger and more complex problems using LSI. Section 4 describes several implementation principles that have proven useful in building LSI-based information systems. Section 5 summarizes some particularly interesting results and surprises that were encountered in the course of building these systems. Section 6 concludes with brief comments on capabilities being incorporated into more recent LSI applications.

3. Enabling Advances in Technology and Technique

3.1. Scaling

Computers in the early days of LSI were not well-suited for SVD computation and large-scale matrix manipulation, which limited the scale of LSI applications. However, hardware improvements over the past twenty years have completely changed this situation [16]. Figure 1 shows the dramatic reduction in the observed index creation times in nine comparable projects over the period 2002-2016. The times shown are those required to build an LSI index for one million documents (averaging several kilobytes in size) at 300 dimensions, using computers typically employed in applications in the given years.

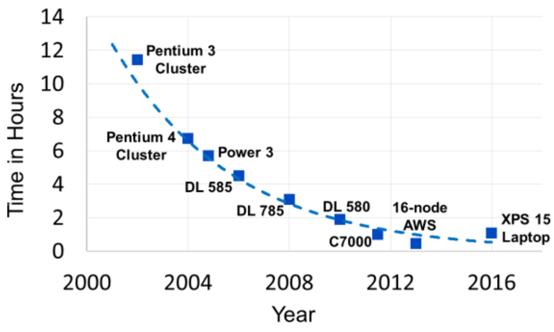


Figure 1. Decline in time required to create an LSI space for a 1 million document collection

The early points on the curve correspond to index creation times from projects using clusters of processors. Subsequent points are from projects primarily employing mid-range servers. Of note, however, the last point shown is for a laptop computer.

The dramatic decline in time required to create an LSI index progressively enabled a wider variety of applications. At the present time, LSI applications involving collections of tens of millions of documents are routine and multiple applications have been implemented that encompass full LSI indexing of hundreds of millions of documents.

Advances in technology enabled improvements not only in scale, but also in the fidelity of the generated LSI spaces in representing real-world semantic associations. For LSI, as collection size increases, the larger number of occurrences of individual terms diminishes the effects of idiosyncratic occurrences of those terms in specific documents. This improves overall representational fidelity, as shown in Figure 2. The graph displays the variation in mean reciprocal rank (MRR) of 250 pairs of terms having known real-world semantic

[2] Other projects undertaken in this time frame applied LSI to data other than text. However, only systems that focused on text are addressed here.

association[3], as a function of the size of the collection. As indicated by the trend line, the increase in representational fidelity with collection size is approximately logarithmic. Of note is the fact that over 80% of all published literature on LSI deals with collection sizes smaller than the initial point shown (17 thousand documents) and 97% deals with collection sizes less than the second point shown (93 thousand documents).

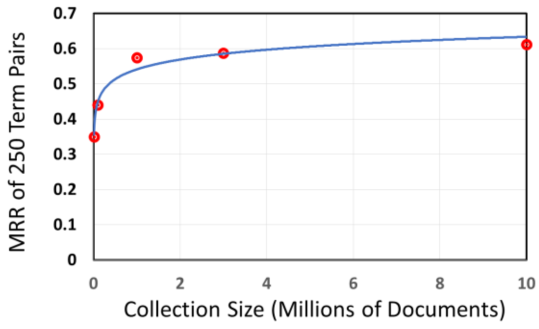


Figure 2. Increase in semantic representation fidelity with collection size

Several of the developed applications included the identification of specific patterns of activities and relationships as one of the system objectives. In many cases, the distinctions between patterns of interest and normal patterns were quite subtle. In general, the larger the data collection, the more effective LSI was in providing indicators of the existence of patterns of interest. Over time, the continuing growth in the size of collections that could be addressed facilitated implementation of increasingly sophisticated analytic operations.

3.2. Parameter Optimization

In the construction of an LSI space, there are a number of parameter choices that must be made; for example: number and identity of stopwords, required number of occurrences for a term to be included in the processing, and the number of dimensions for the LSI space. The choices that are made can have a significant impact on overall performance in a specific application [17, 18]. As an example, Figure 3 shows the variation in mean reciprocal rank of 250 pairs of semantically-related terms as a function of the number of dimensions chosen, for a collection[4] of five million documents [19].

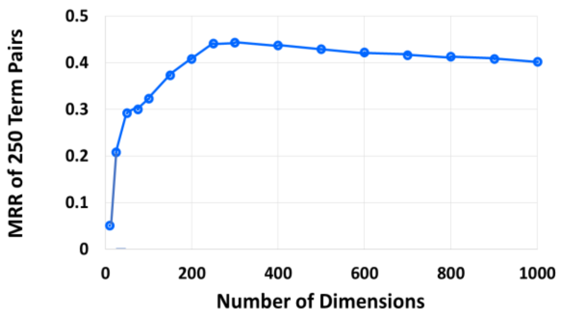


Figure 3. Variation in term similarity ranking as a function of chosen dimensionality

The performance of the systems discussed here benefited greatly from the cumulative experience gained over 20 years regarding choice of effective parameters. In nearly all of the systems developed, at least some testing of parameter choices was carried out that was designed to optimize application performance. This is addressed further in section 4.

3.3. Indexing of Named Entities

In most text applications, named entities constitute items of particular significance. For example, names of people are of fundamental importance in fraud detection. One of the most important factors contributing to the success of the programs described here was the fact that nearly all of them employed entity extraction and markup as a preprocessing step prior to creating the LSI spaces involved. Typically, names of persons, locations, and organizations were extracted, but in some cases more entity types were treated. In the LSI preprocessing, occurrences of a name such as *John Kennedy* were marked up as p_john_kennedy_p, and similarly for other entity types. (This markup was stripped out prior to presenting results to users.)

With classical LSI, users can create queries of the form: *What **terms** are most closely associated with a given **term**?* In contrast, with entity markup prior to creating the space, an interface can be implemented that allows users to enter queries such as: *What **people** are most closely associated with a given **entity** or **activity**?* Such queries are much more natural in most

[3] Over the years, such simple and direct metrics for quality of an LSI space proved to be very useful in tuning implemented systems. Over a wide range of applications, evaluations using such metrics correlated well with both performance on application-specific tests and with human judgment.

[4] These are not the same documents as those in the collection referenced in Figure 2.

applications. The implementation of capabilities to effectively execute those types of queries was a major factor contributing to both operational efficiency and user satisfaction for the systems described here.

Even in the limited number of cases where entities themselves were not of prime importance for users, entity markup prior to creating the LSI space was of great importance for improving the representational fidelity of the space. In most text collections, failure to treat named entities as textual units will create vast numbers of spurious associations. For example, the common English given name *John* may be a component of hundreds of distinct person names. Classical LSI will conflate all of the occurrences of *John*, generating erroneous correlations in the LSI space produced. In many current LSI applications, the text collections being addressed contain millions to tens of millions of named entities. Failing to treat these entities as textual units when building an LSI space for such applications would yield millions of distortions of relations in the space.

3.4. Dealing with Phrases

Many LSI applications involve retrieval of information of interest based on queries formed by users. It is well-known that, in many instances, the use of phrases in queries can significantly aid in expressing a user's information needs. Historically, one frequently-cited criticism of classical LSI was that it did not provide a viable mechanism for dealing with phrases in queries. It was felt that use of phrases required identification of all phrases of interest prior to creating the LSI space, so that those phrases could be treated as terms in the indexing process. This is a problem in that there are a very large number of phrases in a text collection of any significant size. Most of the candidate phrases will never be employed by users. Moreover, indexing of most phrases will not significantly improve the representational fidelity of the LSI space.

We eventually found a two-part solution to this problem. In order to incorporate phrases that would improve the representational fidelity of the space, we employed the following procedure:

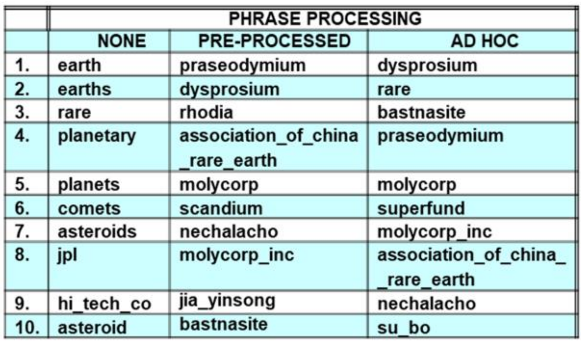
- Using a highly productive phrase generation technique, such as RAKE [20], generate a large set of candidate phrases for the collection of interest.
- Create an initial LSI index for the collection, with no attempt to extract phrases.
- For each candidate phrase, create an approximate LSI vector by taking a weighted average of the representation vectors for the documents that contain that phrase. (The folding-in technique of classical LSI applied to terms [1]).
- Compare the approximate vector for the phrase with a vector created by simply combining the terms of the candidate phrase as an LSI query.
- Create a final LSI space, treating as textual units the candidate phrases which have the greatest distance (smallest cosine) between the approximation vector and the query vector.

In order to ensure that users could employ arbitrary phrases in searches, we developed a technique that allowed use of phrases in LSI queries *even for LSI spaces where phrases have not been indexed.* The technique is described in detail in [21]. Table 1 shows the results from applying this technique in searching a collection of 1.6 million news articles using the query *rare earth element*.

The column labeled NONE shows the ranked query results (closest terms) when no phrase processing is applied. In this case, since the terms *rare* and *element* occur in diverse contexts, the term *earth* has the most significant effect on the results. The results are completely dominated by celestial references; clearly not what a user would desire.

The column labeled PRE-PROCESSED shows the results for the same collection when *rare earth element* was marked up as a phrase and treated as a textual unit in creating the LSI space. The results are as expected: primarily names of rare earth elements and those of people and organizations associated with processing of rare earth elements.

Table 1
Comparison of pre-indexed and ad hoc phrase processing



	PHRASE PROCESSING		
	NONE	PRE-PROCESSED	AD HOC
1.	earth	praseodymium	dysprosium
2.	earths	dysprosium	rare
3.	rare	rhodia	bastnasite
4.	planetary	association_of_china _rare_earth	praseodymium
5.	planets	molycorp	molycorp
6.	comets	scandium	superfund
7.	asteroids	nechalacho	molycorp_inc
8.	jpl	molycorp_inc	association_of_china_ _rare_earth
9.	hi_tech_co	jia_yinsong	nechalacho
10.	asteroid	bastnasite	su_bo

The column labeled AD HOC shows the results when the term folding approach of [21] is applied to the LSI space where there was no initial phrase processing. The results are quite close to those obtained for the case where the phrase was indexed (60% agreement for the top ten terms in a collection comprising 1.5 million terms). The adoption of this ad hoc phrase query process in systems described here resulted in a major improvement in user satisfaction.

3.5. User Aids

Over the years, with the dramatic growth in the size of the text collections being addressed, it became increasingly important to provide aids for users in areas such as creating queries, identifying topics, interpreting results, and automating repetitive tasks. The semantic comparison capabilities of LSI allowed a wide variety of such aids to be implemented. Some aids were very simple to implement, but still yielded significant gains in operational efficiency and user satisfaction. For example, a popup display of the most closely associated terms when a user moused over a given term was of great help in determining the meaning of newly-encountered terms such as acronyms and technical terminology. Most of the users of the systems were knowledge workers, but typically did not have technical backgrounds. Providing them with immediate contextual information regarding technical terms greatly aided them in understanding the material that they were working with.

Over time, such aids became more complex. One that proved very popular was novelty detection. Within some systems, tracking capabilities were implemented to provide an indication of what information a given user already was aware of. This included, for example, monitoring what documents (or other text objects) that the user had previously displayed, saved, printed, or incorporated into work products. Then, in response to a query from that user, the results could be displayed not just in relevance order, but in the order of those results that were relevant *but at the same time* were least similar to those previously seen. In many applications there is significant redundancy in the content of items collected. In applications with high information redundancy, the novelty detection feature greatly improved both efficiency of operations and user satisfaction.

Other user aids that proved to enhance both operational efficiency and user satisfaction included:

- Generation of document summaries tailored to users' interests.
- Automated generation of graphs showing relationships among entities.
- Automated tracking of topic threads in long documents and sets of documents.

3.6. Secure Information Sharing

The representation for a given term in an LSI space is a single point in a vector space that is derived from what may be hundreds of occurrences, even for a relatively rare term. Similarly, the representation for a given document is derived from large numbers of occurrences of multiple terms. Even in classical LSI spaces, it is impossible to work backwards to reconstruct the actual wording of documents corresponding to extant document vectors. With slight modifications to the index creation process, it can be made impossible to determine even which words occurred in which documents. These characteristics enable the use of information in a secure background mode.

In many applications there is relevant data available that cannot be directly shared with users for proprietary, legal, or privacy reasons. In such cases, these sensitive documents can be processed so that the results of operations in the LSI space for the application can be enhanced by the *contextual implications* of the sensitive data, without risk of disclosure of specific sensitive data items *themselves*.

Experience in using LSI in a secure background mode has shown that even a small number of documents used in this manner can have great leverage. In some representative cases, data treated in background mode has constituted less than 1% of the total data being examined. Nevertheless, significant gains in application efficiency still have been achieved [22].

4. Beneficial Implementation Practices

Over the years, a number of LSI implementation practices evolved that significantly improved the quality and efficiency of the systems developed.

Perhaps the most significant implementation approach adopted was to use analyses in the LSI spaces themselves to select effective values for all of the key processing parameters for an application. Typically, we used the following approach:

1. Build an initial LSI space from application-relevant data, using standard parameter values and processing choices.
2. Using a small, representative test set, carry out analyses in this initial space to determine the most effective values for the parameters and choices.
3. Re-build the LSI spaces using those parameters and processing choices.

For example, using a test set representative of an application being addressed, it is possible to make an effective choice of the number of dimensions to employ in creating the LSI space for that application. As long as the initial space employs a number of dimensions higher than optimal, the requisite tests can be carried out, and an optimal value found, with vectors from a single initial LSI space.

For other parameter choices, a new LSI space must be created to test each value. For example, in many applications, terms are only included in the LSI processing if they occur at least M times in the collection and/or in at least N different documents. Pruning the term set in this manner often can significantly improve the representational fidelity of the space. A separate LSI space must be generated in order to test each prospective pruning value. However, only a limited range of values must be tried. In most of the applications here, values of M and N in the range of two to five turned out to be optimal. It should be noted that pruning typically was not applied to named entities. In many applications, the occurrence of a name may be of significance even if it occurs only once.

The dramatic reduction in the time required to create LSI spaces made it increasingly feasible to create trial LSI spaces for optimization testing, even for parameters that required multiple such spaces to be created. For very large collections, optimization analyses typically can be carried out sufficiently effectively using LSI spaces built from a randomly selected subset of the overall collection.

We also employed iterative refinement of LSI spaces to mitigate the effects of errors in training data for categorization applications. This approach led to significant improvement in categorization accuracy. The technique has broad applicability for noise mitigation in LSI applications [23].

The computer employed to carry out analytic operations in an LSI space does not have to be the same computer on which the LSI space is created. It often proved useful to create LSI spaces on a large server and then distribute the vector spaces created there to smaller devices for use. We also found that distribution of shared LSI spaces can be a powerful enabler for collaborative work.

Sometimes a conceptual search will retrieve results that do not appear to be appropriate. Users may find this disconcerting. However, these often can be the most important results – ones that indicate a gap in user understanding of some aspect of the problem at hand. In multiple systems we found it useful to highlight terms and passages in retrieved documents based on semantic similarity to the user's query. Users found this useful in trying to determine why a surprising result was obtained.

Other implementation principles that proved effective included:

- Duplicate and near-duplicate documents in a collection artificially magnify associated term relationships. LSI comparisons between documents of a collection can be used very effectively to eliminate redundant documents.
- For some applications, removal of "boilerplate" text can greatly enhance performance. For example, many legal documents contain formulaic blocks of text that appear on many documents. Appearance of such repeated text creates undesired associations (i.e., ones that are not related to the *content* of the documents).
- In many instances it is useful to use LSI similarity comparisons to decompose long documents into conceptually cohesive segments, which are then indexed as individual items. This makes it much easier to identify information on subsidiary topics.
- For large applications, parallel processing approaches such as MapReduce and more recent techniques can be employed very effectively for text preprocessing tasks.
- In analyses involving the LSI vectors of large collections, use of GPUs for the cosine comparisons can provide a dramatic speedup compared to using typical CPUs.
- In many applications, entity-driven analytic processes can be far more efficient than document-driven ones.

- Monitoring of user actions often can provide training data that can be employed to refine the LSI spaces employed and to yield improved accuracy of analytic operations. One particularly effective use of this techniques was in continuously refining textual representations of user interests.

5. Interesting Results and Surprises

Over the past 20 years there were a number of aspects of LSI that either came as a surprise or were unexpectedly useful.

When the work described here began, it was generally believed that LSI did not scale well. Academic papers of the time estimated that the time required to build an LSI space grew as at least the square of the number of documents addressed.[5] We were pleasantly surprised that actual measurements showed that the growth was close to linear [16].

Indications of semantic similarity as provided by LSI turned out to be a remarkably good proxy for similarity judgments generated by people. In 2007 a review of 30 studies compared LSI and human judgment in 16 real-world text processing tasks ranging from synonym matching to psychological assessment. LSI performed as well as, or better than, humans in 51% of the cases [5]. In more recent, work, covering over 100 studies and 37 applications, LSI performed as well as, or better than, humans in 56% of the cases [24]. Of significance is the fact that all of these studies employed straightforward implementations of LSI. None of the advanced techniques described in this paper were used in any of the analyzed studies. Moreover, the number of documents used to create the spaces was very small - having a median value of only 1700. With larger collections, LSI performance in the reviewed studies likely would have been significantly higher. In the 63 information systems considered here, in the few cases where human and LSI performance could be directly compared, LSI results typically were as good as, or in some cases somewhat better than, average human performance.

One surprise was the huge effect that treating named entities as textual units produced. For collections of text such as news articles, the representational fidelity of the spaces produced was dramatically improved. Having the entities available also set us on a path of implementing ever more sophisticated entity-driven analysis capabilities. In most applications, entity-driven processes turned out to be far more efficient that document-driven ones.

Many of the applications addressed were complicated by the fact that the text items of interest contained multiple variants of names of individuals. These differences came from misspellings, phonetic renderings, transliteration differences, and other sources. Because of these variations, many relationships of interest were suppressed. One of the early features that we implemented was a name variant analyzer. For any given name it combined eight methods for generating candidate variants and then used comparisons in the LSI space to select the most relevant ones. This capability turned out to be significantly more effective than the best competing commercial product. Recall was two to three times greater and confidence ratings for candidate equivalent names turned out to be much more reliable than anticipated [25].

We were surprised by how easy it was to implement ad hoc phrase processing in LSI spaces. (We also were embarrassed by how long it took for us to realize how to do it).

It was interesting to observe how easily and effectively word senses could be disambiguated using clustering techniques in the LSI spaces [26]. This allowed markup of occurrences of polysemous words in much the same way as was done for named entities, as was described in section 3.3. The disambiguation can be carried out in a trial space and then the marked-up senses of polysemous words treated as separate textual units in creating the final space to be employed. Typically, a point of diminishing returns will be reached after disambiguating only a few thousands to tens of thousands of words. For some applications, word sense disambiguation of general terms did not result in major performance increases. Where disambiguation was of great value, however, was in dealing with person names. In many applications there may be hundreds of people with the same name and disambiguation is essential. As with phrases, this name resolution feature can be incorporated into

[5] Early estimates tended to overlook one or more of three key factors. First, LSI requires calculation of only the first few hundred singular values and associated vectors, not a complete SVD of the entire term-document matrix. Second, term-document matrices are extremely sparse. For large collections, often only one in ten thousand to one in one hundred thousand entries is non-zero. Finally, the time required to read and preprocess the text being indexed generally is greater than the time required to carry out the SVD.

an application either in bulk during preprocessing or in an ad hoc fashion at query time.

Applications involving the secure background mode of dealing with sensitive data often involved very small amounts of such data (sometimes less than .01% of the total amount of data). In a number of cases, the extent to which such very small amounts of auxiliary data could improve results was quite remarkable.

Some of the early applications involved text that was produced by optical character recognition (OCR) equipment. LSI turned out to be surprisingly effective in dealing with the many errors produced by OCR devices of that era. In one categorization application, performance degradation only began to be detectable when the OCR error rate reached a level where two out of every three words were corrupted [27].

In cross-lingual applications, it turned out that many languages can be represented in a single LSI space without serious performance degradation. In one case, transitioning from two languages represented in one LSI space to 13 languages resulted in a decline in cross-lingual similarity comparisons of only a few percent [28].

LSI turned out to provide an elegant solution for combining results from diverse information systems when employing federated queries [29].

Combining text with other data types (especially relational, geographic, and image data) often generated unique analytic insights. The combination of such data also supported implementation of highly effective visual analytic interfaces.

6. Recent Developments

Although much has been accomplished over the past twenty years, there are still exciting activities underway involving LSI. Many of these involve implementation of ideas that were originally suggested in a basic form some years ago, but are just now being incorporated into real-world applications. Key examples include:

- Combined analysis of text and relational data [29].
- Implementation of semantic vector space equivalents of Boolean operators [33] and negation [34].
- Enhancement of machine translation capabilities, especially for technical and other specialized subject matter [38].
- Functionality based on analysis of individual LSI vector components.[6] [30, 31, 32].
- Use of randomized SVD to dramatically reduce the computational load when addressing very large collections [35, 36, 37].
- Extensive use of LSI in discovery applications, particularly in the area of bioinformatics [12].
- Facilitation of human-robot interaction [39, 40].
- Various AI-related efforts [41,42].

7. Acknowledgements

I would like to thank the engineers, scientists, software developers, and others from SAIC, Content Analyst Company, and Agilex Technologies Inc. who participated in building the systems reviewed here over the past twenty years. Those individuals took the techniques described here for improving LSI and transformed them from nascent concepts into working code and deployed systems.

8. References

[1] George W. Furnas, et al. Information retrieval using a singular value decomposition model of latent semantic structure, in: Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 88), May 1988, Grenoble, France, pp. 465–480.

[2] Susan T. Dumais, Latent Semantic Analysis. Annual Review of Information Science and Technology, 38(1), 2004, pp. 188-230, doi:https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/aris.1440380105.

[3] Jerome R. Bellegarda, Latent Semantic Mapping: Principles & Applications. Morgan & Claypool, 2007 doi: https://doi.org/10.2200/S00048ED1V01Y200609SAP003.

[6] The components of a typical LSI vector comprise hundreds of indications of derived relationships. In general, the basis vectors of an LSI space closely relate to concepts, or mixtures of such, within the collection of text being addressed. An LSI vector is thus a complex and information-rich object. To compare two such objects using a single number (such as a cosine) thus ignores a large amount of potentially useful information.

[4] Thomas K. Landauer, Danielle S. McNamara, Simon Dennis, and Walter Kintsch, eds. 2007. Handbook of Latent Semantic Analysis. Lawrence Erlbaum Associates.

[5] Roger Bradford, Comparability of LSI and human judgment in text analysis tasks, in: Proceedings, Applied Computing Conference, September 28-30, 2009, Athens, Greece, pp. 359-366.

[6] Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman, Using latent semantic analysis to improve access to textual information, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, May 1988, Washington, DC, pp. 281-285.

[7] Tristan Miller, Essay assessment with latent semantic analysis, Journal of Educational Computing Research, 29(4), (2003) 495-512.

[8] Lexis-Nexis, The Evolution of Semantic Search on the Web, 2009. URL: https://www.lexisnexis.co.uk/pdf/brochures/totalpatent-whitepaper.pdf.

[9] Jean Isson, Unstructured Data Analytics: How to Improve Customer Acquisition, Customer Retention, and Fraud Detection and Prevention, John Wiley & Sons, 2018.

[10] Seraina Anagnostopoulou, et al., The impact of online reputation on hotel profitability, International Journal of Contemporary Hospitality Management September 20, 2019. doi: 10.1108/IJCHM-03-2019-0247.

[11] Wei Dong, et al., The detection of fraudulent financial statements: an integrated language model, in: Proceeding of the 19th Pacific-Asia Conference on Information Systems (PACIS 2014), Article 383.

[12] Roger Bradford, An overview of information discovery using latent semantic indexing, in: Proceedings, International Conference on Computer Science, Applied Mathematics and Applications (ICCSAMA 2017), June 30 -July 1, 2017, Berlin, Germany, pp. 153-164.

[13] Hongyu Chen, et al., Effective use of latent semantic indexing and computational linguistics in biological and biomedical applications, Frontiers in Physiology, 4: 8. (2013) doi: 10.3389/fphys.2013.00008.

[14] Nguyen Ngoc Chan, Walid Gaaloul, and Samir Tata, A web service recommender system using vector space model and latent semantic indexing, in: Proceedings, 2011 IEEE International Conference on Advanced Information Networking and Applications, March 22-25, 2011, Biopolis, Singapore, pp. 602-609.

[15] T. Hashimoto, T. Kuboyama and B. Chakraborty, Temporal awareness of changes in afflicted people's needs after East Japan Great Earthquake, in: Proceedings, IEEE International Conference of IEEE Region 10 (TENCON 2013), 1-6. doi: 10.1109/TENCON.2013.6719012.

[16] Roger Bradford, Implementation techniques for large-scale latent semantic indexing applications, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM), October 2011, Glasgow Scotland, pp. 339-344.

[17] Zhiqiang Cai et al, Impact of corpus size and dimensionality of LSA spaces from Wikipedia articles on AutoTutor answer evaluation, in: Proceedings, 11th International Conference on Educational Data Mining (EDM), Jul 16-20, 2018, Raleigh, NC, pp.127-136.

[18] Thomas K. Landauer and Susan T. Dumais, A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological Review, 104(2), (1997) 211-240.

[19] Roger Bradford, An empirical study of required dimensionality for large-scale latent semantic indexing applications, in: Proceedings of the 17th ACM conference on Information and knowledge management (CIKM 2008), October 19-23, 2008, Napa Valley, CA, pp. 153-162. doi: https://doi.org/10.1145/ 1458082.1458105.

[20] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley, Automatic keyword extraction from individual documents. Text Mining: Applications and Theory 1 (2010): 1-20.

[21] Roger Bradford, Incorporating ad hoc phrases in LSI queries, in: Proceedings, 6th International Conference on Knowledge Discovery and Information Retrieval, October 21-24, 2014, Rome, Italy, pp. 61-70.

[22] Roger Bradford, Exploiting sensitive information in background mode using latent semantic indexing, in: Proceedings of the Sixth Workshop on Link Analysis, Counterterrorism and Security, SIAM Data Mining Conference, April 24-26 2008, Atlanta, Georgia.

[23] Roger Bradford and John Pozniak, A systematic approach to design of a text categorizer, in; Proceedings, 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), October 9-12, 2016, Budapest, Hungary, pp. 509-514.

[24] Roger Bradford, Comparability of LSI and human judgment in text analysis tasks – an update. Draft, Mar 2, 2021.

[25] Roger Bradford, Use of latent semantic indexing to identify name variants in large data collections, in: Proceedings, 2013 IEEE International Conference on Intelligence and Security Informatics (ISI 2013), Seattle, WA, 27-32. doi: 10.1109/ISI.2013.6578781.

[26] Roger Bradford, Word sense disambiguation, 2008. Patent No. 7,415,462, Filed January 20, 2006, Issued August 19, 2009.

[27] Anthony Zukas and Robert Price, Document categorization using latent semantic indexing, in: Proceedings, Fifth Annual Symposium on Document Image Understanding Technology (SDIUT), April, 2003, Greenbelt, MD, pp. 87-91.

[28] Roger Bradford and John Pozniak, Combining modern machine translation software with LSI for cross-lingual information processing, in: Proceedings, 11th International Conference on Information Technology: New Generations (ITNG), April 7-9, 2014, Las Vegas, NV, 65-72. doi: 10.1109/ITNG.2014.52.

[29] Roger Bradford, Federated queries and combined text and relational data, 2006. Patent Application No. 11434749, Filed May 17, 2006.

[30] Weizhong Zhu, and Chaomei Chen, Storylines: Visual exploration and analysis in latent semantic spaces. Computers & Graphics, 31(3) (2007): 338-349.

[31] Ricardo Olmos, et al, Transforming selected concepts into dimensions in latent semantic analysis, Discourse Processes, 51(5-6) (2004): 494-510.

[32] Anna Sidorova, Nicholas Evangelopoulos, Joseph S. Valacich, and Thiagarajan Ramakrishnan, Uncovering the intellectual core of the information systems discipline. MIS Quarterly (2008): 467-482.

[33] Preslav Nakov, Getting better results with latent semantic indexing, in: Proceedings of the Students Presentations at the 12th European Summer School in Logic, Language and Information (ESSLLI), August 6-18, Birmingham, UK, pp. 156-166.

[34] Dominic Widdows, Orthogonal negation in vector spaces for modelling word-meanings and document retrieval, in: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, July 7, 2003, Volume 1, pp. 136-143.

[35] Nathan Halko, Per-Gunnar Martinsson, and Joel Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Review, 2(3) (2011): 217-288.

[36] Ming Gu, Subspace iteration randomization and singular value problems, SIAM Journal on Scientific Computing, 37(3) (2015): 1139-1173.

[37] Per-Gunnar Martinsson, Randomized methods for matrix computations, in: The Mathematics of Data, IAS/Park City Mathematics Series, 25(4) 2018, pp. 187-231.

[38] Roger Bradford, Machine translation using vector space representations, 2010. Patent No. 7,765,098, Filed April 24, 2006, Issued July 27, 2010.

[39] Phoebe Liu, Dylan Glas, Takayuki Kanda, and Hiroshi Ishiguro, Data-driven HRI: Learning social behaviors by example from human–human interaction, IEEE Transactions on Robotics, 32, no. 4, (2016): 988-1008.

[40] Francesco Agostaro, et al., A conversational agent based on a conceptual interpretation of a data driven semantic space, in: Proceedings, Congress of the Italian Association for Artificial Intelligence, September 21–23, 2005, Milan, Italy, pp. 381-392.

[41] Robert Speer, Catherine Havasi, and Henry Lieberman, AnalogySpace: Reducing the dimensionality of common-sense knowledge, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Jul 13, 2008, vol. 8, pp. 548-553.

[42] Trevor Cohen, Brett Blatter, and Vimla Patel, Simulating expert clinical comprehension: Adapting latent semantic analysis to accurately extract clinical concepts from psychiatric narrative, Journal of Biomedical Informatics 41, no. 6 (2008): 1070-1087.