Instability of Relevance-Ranked Results Using Latent Semantic Indexing for Web Search

Houssain Kettani
ECECS Department
Polytechnic University of Puerto Rico
San Juan, PR 00919
hkettani@pupr.edu

Gregory B. Newby
Arctic Region Supercomputing Center
University of Alaska
Fairbanks, AK 99775
newby@arsc.edu

Abstract

The latent semantic indexing (LSI) methodology for information retrieval applies the singular value decomposition to identify an eigensystem for a large matrix, in which cells represent the occurrence of terms (words) within documents. This methodology is used to rank text documents, such as Web pages or abstracts, based on their relevance to a topic. The LSI was introduced to address the issues of synonymy (different words with the same meaning) and polysemy (the same words with multiple meanings), thus addressing the ambiguity in human language by utilizing the statistical context of words. Rather than keeping all k *possible eigenvectors and eigenvalues from the singular value decomposition which approximates the original term by document matrix, a smaller number is used – essentially allowing a fuzzy match of a topic to the original term by document matrix. In this paper, we show that the choice* k *impacts the resultant ranking and there is no value of* k *that results in stability of ranked results for similarity of the topic to documents. This is a surprising result, because prior literature indicates that eigensystems based on successively large values of* k *should approximate the complete (max* k*) eigensystems. The finding that document-query similarity rankings with larger values of* k *do not, in fact, maintain consistency, makes it difficult to assert that any particular value of* k *is optimal. This in turn renders LSI potentially untrustworthy for use in ranking text documents, even for values that differ by only 10% of the max* k.

This work was supported in part by the Arctic Region Supercomputing Center at the University of Alaska – Fairbanks, with additional support from the US Department of Defense High Performance Computing Modernization Program Office under contract G00003435 and others. This work was also supported in part by the United States Nuclear Regulatory Commission under grant number NRC-27-09-310.

1. Introduction

Searching large collections of documents is challenging for computational, practical and humanistic reasons. The great success of today's online search engines has allowed us to overlook many of these challenges, since results are, generally, satisfying. Yet, only a small fraction of available electronic data is available through such search engines [8, 12]. From a practical standpoint, we would like to make increased numbers of items of all types (Web pages and other documents) available for search. In some cases, this will require interacting directly with a search service on a specific site, rather than a centralized aggregation of content (as seen in Google & others). Such distributed search might offer the only method for overcoming security and accessibility concerns for access to specific data sets [5].

If practical considerations for distributed search were overcome, there would still be computational and humanistic concerns remaining. Humanistically, consider that the role of search systems – more generally called information retrieval or IR systems – is to provide a ranked list of the most relevant documents. This list is created based on matching the human expression of information need – the query – to a set of documents. The set of documents can be very large, as we see with today's Web search engines. The challenge is that relevance to a human's information need is a difficult concept [10]. The list of relevant documents for a query is situational, and for a particular human information seeker the relevance of a given document will depend on what he or she is really looking for, as well as prior knowledge and expertise in the subject area. Expecting an IR system to make an effective ranked list of documents based on just a few query words is ambitious, indeed! Successful search engines today often present a variety of different results in the first set of 10 or 20 documents, so that different information seekers can quickly find high-

quality documents of interest in the list. In this paper, we discuss latent semantic indexing (LSI), which has been proposed as one approach to better match the conceptual meaning of a query against the concepts found in a document, rather than being solely reliant on the (few) terms of a query, and the often ambiguous language used in queries and documents [2].

Computational challenges of search have, to some extent, been addressed through heroic engineering undertaken by database and Web search engine providers, among others. Being able to respond to a query in only a second or so is testament to the ability to rapidly produce results by parallelizing search across a single large corpus. In this paper, we consider the computational challenges brought about by distributed search and the merger of results from multiple collections, which has not yet been the subject of intense engineering efforts. We anticipate that the practical need to search across data sets, along with the humanistic need to get search results which are more closely targeted for individual information needs, will provide impetus to address such problems.

In this paper, we describe the LSI approach to information retrieval. We compare its successive application by incorporating different numbers of eigenvalues k that differ by 10% of the maximum k. We develop a statistical approach to compare the resultant ranking from each k value. We observe that the rankings differ significantly. The finding that document-query similarity rankings with larger values of k do not, in fact, maintain consistency, makes it difficult to assert that any particular value of k is optimal. This in turn renders LSI potentially untrustworthy for use in ranking text documents, even for values that differ by only 10% of the max k.

The rest of the paper is organized as follows. The vector space model and latent semantic indexing algorithms are described in Sections 2 and 3. A discussion of various performance challenges for LSI is laid out in Section 4. Accordingly, Section 5 presents an empirical evaluation of the robustness of LSI with respect to the choice of k. Finally, concluding remarks are presented in Section 6.

2. Vector Space Model

When addressing the issue of document ranking, it is convenient to reduce it to a mathematical problem. Thus, [9] introduced the vector space model (VSM), in which keywords (terms) are the axis and documents are points in a multidimensional space. For example, if we have two documents $\mathbf{d_1}$ and $\mathbf{d_2}$, the keywords are the set {cat, dog}, the word "cat" occurs five and seven times in $\mathbf{d_1}$ and $\mathbf{d_2}$, respectively, and the word "dog" occurs two and three times in $\mathbf{d_1}$ and $\mathbf{d_2}$, respectively, then $\mathbf{d_1}$ and $\mathbf{d_2}$ are represented by the points (5, 2) and (7, 3), respectively, and the query vector could be (0, 1), (1, 0), or (1, 1). A pictorial illustration of this vector space model is given in Figure 1.

The ranking of the similarity of all documents to the query is achieved by calculating the distance between the documents and the query, then ranking all documents based on their distance measure. The closer a document to a query is, the higher its rank. Alternatively, since calculating distances is time consuming, we can use the angle θ_i between each document and the query as a measure for ranking. More precisely, we can use the cosine of this angle for ranking. Thus, we can define,

$$\alpha_i = \cos\,\theta_i = \frac{\mathbf{q}^T \mathbf{d}_i}{\| \mathbf{q} \| \| \mathbf{d}_i \|}.$$

Furthermore, if the vectors are normalized to unit length, this measure can be reduced to $\alpha_i = \mathbf{q}^T\mathbf{d_i}$. Thus, if all vectors are normalized, and since all the points lie in the nonnegative hypercube, minimizing the distance between $\mathbf{d_i}$ and $\mathbf{q}$ is equivalent to maximizing α_i, which is a simple dot product between the query and the corresponding document. Additional computational savings are gained by the fact that any document that does not contain any query terms will have a cosine of zero, thus immediately eliminating such document from further consideration.

The VSM was introduced as an alternative to an even simpler approach called Boolean matching. With Boolean matching, any document with all query terms is considered equivalently relevant to any other document with all query terms. While this approach was suitable in the past for small collections, and for long queries, it is clearly inappropriate for today's collections of millions or billions of documents. The tendency of many queries to have very few words makes the problem worse. With the VSM, document length and query term frequency are taken into account, such that the relative importance of query terms within each document is taken into account. Other techniques, such as pivoted document length normalization, enhanced the VSM further.

The vector space model and follow-on techniques is an excellent approach to information retrieval, and scales well to larger collections. In fact, VSM is at the heart of most IR systems today. However, the VSM does not adequately address the issues of synonymy and polysemy of words in a language, such as "car" vs. "automobile," or the use of "can" in "I can" versus "can" in "tuna can." Nevertheless, its application is relatively fast. Accordingly, obtaining each α_i requires $O(n)$ multiplications (where n is the number of documents from a corpus that contain at least one query term). Therefore, obtaining the score vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$ requires $O(n^2)$, which is the time complexity of VSM. This can further be reduced to $O(n)$ through parallelization.

3. Latent Semantic Indexing

The latent semantic indexing (LSI) algorithm [2] was introduced as an enhancement to the vector space model to address the issues of synonymy and polysemy through statistical treatment of the relations among terms and documents. LSI applies the singular value decomposition (SVD) to reduce linear redundancies in data. Accordingly, [2] points out that a corpus can be presented as a matrix $\mathbf{C}$ with non-negative integer entries representing the number of occurrence of each term in each document. The columns of $\mathbf{C}$ represent documents and its rows represent the terms as follows:

$$\mathbf{C} = \overbrace{\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & \ddots & & \\ \vdots & & \ddots & \\ c_{n1} & c_{n2} & \cdots & c_{nm} \end{bmatrix}}^{m \text{ documents}} \Big\} n \text{ terms}$$

For a reasonably sized test collection, such as the Web2 collection from the annual Text REtrieval Conference (http://trec.nist.gov), m is on the order of millions, while n is hundreds of thousands. Latent semantic indexing suggests first obtaining the corresponding singular value decomposition as follows:

$$\mathbf{C} = \mathbf{USV}^T = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & \ddots & & \\ \vdots & & \ddots & \\ u_{n1} & u_{n2} & \cdots & u_{nm} \end{bmatrix} \begin{bmatrix} s_1 & & & 0 \\ & s_2 & & \\ 0 & & \ddots & \\ & & & s_m \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & \ddots & & \\ \vdots & & \ddots & \\ v_{m1} & v_{n2} & \cdots & v_{mm} \end{bmatrix}^T$$

Then, a rank reduction is obtained by zeroing out the singular values s_i such that $s_i << s_1 \ \forall \ i \geq k$, which introduces the following approximation of the term-by-document matrix $\mathbf{C}$:

$$\hat{\mathbf{C}} = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^T = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & \ddots & & \\ \vdots & & \ddots & \\ u_{n1} & u_{n2} & \cdots & u_{nk} \end{bmatrix} \begin{bmatrix} s_1 & & & 0 \\ & s_2 & & \\ 0 & & \ddots & \\ & & & s_k \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & \ddots & & \\ \vdots & & \ddots & \\ v_{m1} & v_{n2} & \cdots & v_{mk} \end{bmatrix}^T$$

After normalizing the columns of $\hat{\mathbf{C}}$, the vector space model approach of taking the cosine between query vectors and document vectors can be applied. An itemized description of the LSI algorithm is as follows. We start by using VSM to represent the query and corpus by letting:

1. $\mathbf{q}$ represent an $n \times 1$ query vector with unit norm, and
2. $\mathbf{C}$ represent an $n \times m$ term by document matrix.

We then apply LSI as follows:

3. $\mathbf{C} = \mathbf{USV}^\mathbf{T}$ is the SVD of $\mathbf{C}$,
4. Obtain $\hat{\mathbf{S}}$ by keeping the first k diagonal elements of $\mathbf{S}$,
5. Adjust $\mathbf{U}$ and $\mathbf{V}$ accordingly to get $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$, by keeping the first k columns of each,
6. Approximate $\mathbf{C}$ by $\hat{\mathbf{C}} = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}$,
7. Normalize the columns of $\hat{\mathbf{C}} = [\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2, \ldots, \hat{\mathbf{c}}_m]$,
8. Define $\alpha_i = \mathbf{q}^\mathbf{T}\hat{\mathbf{c}}_\mathbf{i}$,

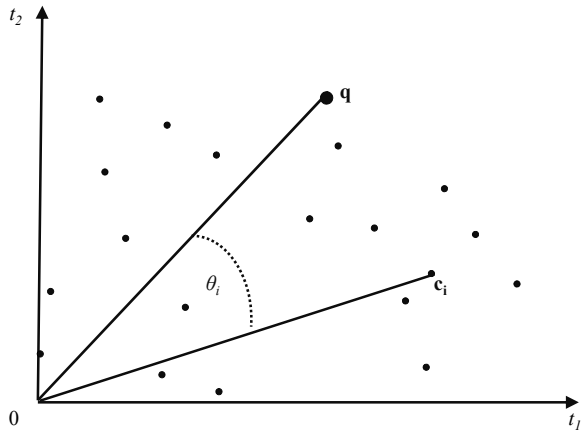

Figure 1. Pictorial illustration of the vector space model of document ranking, where $\{t_1, t_2\}$ is the set of keywords, q is the query, other points are the documents, θ_i is the angle between a document and the query.

9. Let $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$ be the score vector,
10. The higher the score is, the higher the rank of the document.

Consequently, the time complexity of LSI can be obtained as follows. The singular value decomposition of the $n \times m$ matrix **C** in Step **3**, requires $O(nm^2 + n^2m + n^3)$ multiplications [13]. The multiplication of an $n \times m$ matrix **A** and an $m \times p$ matrix **B** requires $O(nmp)$ [13]. Thus, obtaining the matrix $\hat{\mathbf{C}}$ in Step **6**, requires $O(\max\{nk^2, nkm\}) = O(nkm)$, since $k < m$. On the other hand, obtaining each α_i in Step **8**, requires $O(n)$, and therefore, obtaining the score vector $\boldsymbol{\alpha}$ in Step **9**, requires $O(n^2)$, which is the time complexity of VSM. Since LSI consists mainly of the aforementioned three parts, then it can readily be seen that SVD controls the complexity of LSI. Accordingly, the time complexity of LSI algorithm is in $O(nm^2 + n^2m + n^3)$.

4. Performance Issues

For a very large corpus, LSI is intractable with current microprocessors and memory constraints. For example, the TREC Million Query Track collection [11], results in a matrix **C** with n of about 600,000 terms by m of about 25 million documents. Storing this matrix is doable, since it is very sparse with about 99.99% of its elements bring zero. (This is because most terms do not occur in most documents.) However, TREC participants have not yet produced an SVD for this matrix, even on today's large supercomputers. This motivated researchers such as [14], to parallelize the problem to better utilize the rapidly advancing high performance computing infrastructures.

Singular value decomposition (SVD) produces a diagonal matrix containing singular values of the matrix **C** in descending order. As mentioned earlier, in its attempt to approximate **C**, LSI zeroes out the singular values s_i such that $s_i << s_1 \; \forall \; i \geq k$. But when is the condition "$s_i << s_1$" satisfied? In his illustrative example, [2] picked the largest two singular values and zeroed out the other seven. So the authors suppressed all singular values less than 71% of the largest singular value. Others, such as [3], were more cautious and argued that the condition "$s_i << s_1$" corresponds to s_i less than 10% of s_1.

To empirically evaluate the performance of a document ranking algorithm, it is tempting to use random matrices to simulate the term-by-document matrix **C**. This randomness, however, makes singular values decrease very slowly even if we ensure that the matrix is very sparse. For example, we generated sparse random matrix in MATLAB with the following: "**C** = ceil(r – rand(m, n))," where $0 \leq r \leq 1$. This results in about $100r$% of the elements of **C** being nonzero. The slow decay of singular values will entail keeping most of them. For instance, if the 10% criterion is followed, we will need to keep 75% to 99% of the singular values. This in turn, defeats the rationale behind the use of LSI.

Nonetheless, the structure of human languages makes the content and formation of a text document far from being random. To the contrary, LSI is designed to make use of the non-randomness of human language. For our purposes, the TREC Million Query Track [11] provides an excellent, yet challenging, datasets for evaluation purpose. This dataset consists of millions of English Web documents and queries gathered from the United States .gov domains.

We constructed **C** from sub-collections of this dataset based on separate specific domains (such as www.nih.gov or www.whitehouse.gov), each with thousands of unique terms and documents. More than 99% of the elements of **C** in these sub-collections were zero. We observed an exponential decrease in the magnitude of the corresponding singular values. For example, following the 10% criterion, we keep only 0.15% to 0.69% of the singular values.

Finally, it is important to note that in a document ranking algorithm, the order of documents in a corpus should not affect their ranking. Indeed, if SVM is applied alone, then the partitioning does not affect the ranking. However, due to LSI, we generally have $\hat{\hat{\mathbf{C}}} \neq \hat{\mathbf{C}} \neq \mathbf{C}$. Thus, in the next section; we investigate the stability and robustness of LSI using the TREC dataset.

5. Empirical analysis

5.1. Real data

To evaluate the performance of LSI, we applied it to sub-collections of the corpus obtained from the TREC Million Query Track [11]. TREC is the Text REtrieval Conference, held annually at the National Institute for Standards and Technology (NIST) in Gaithersburg, Maryland. TREC is intended to provide a forum for comparison of information retrieval techniques across a variety of test conditions. Different tasks are evaluated as part of "tracks," which typically last for at least a few years before being updated, merged with other tracks, or discontinued. In the past, TREC focused on relatively structured textual data, such as abstracts, news articles and proceedings such as the Federal Register.

The growth of the Web, and interest in Web-based search engines, has resulted in numerous tracks that examined HTML, often harvested from the live Web. These harvests resulted in fixed test collections, with the largest in 2008 holding over 20 million unique Web pages from over 6,000 unique hosts. By reusing a test collection for multiple years of TREC experiments, a set of judgments for the relevance of series of queries (known in TREC as topics) against the collection (known as a corpus) can help to build knowledge of the collection, and compare results from one year to other years.

At the Arctic Region Supercomputing Center (ARSC), we have participated in TREC's very large corpus track (VLC), million query track (MQ) and terabyte track (TB) in past years. These tracks have provided tens of thousands of unique TREC topics for evaluation against subsequently larger collections of Web-based documents.

In [4], we used an early approach to the divide and conquer method described in [14]. This followed past experiments based on techniques from [1] and other methods for merging results from separate systems. For the 2008 TREC million query track, our approach was to portray collections (which were sub-collections of the entire corpus, based on specific Web sites within .gov) as vectors in an LSI space. For a given query, we determined the most similar collections, based on the cosine between the collection vectors and query vectors. The fifty collections with the highest cosines were then searched by LSI using the algorithm as described above, rather than the whole set of over 6,000 collections.

By applying LSI techniques at the collection level, to rank and select collections prior to LSI at the document level, we were able to obtain greater efficiency by selecting only those collections deemed to be more highly related to a query. For practical purposes with distributed search, this is important since the incremental cost for adding more collections (in terms of latency and larger sets of documents to rank) can be avoided by only searching collections of greater interest.

5.2. Performance evaluation

From TREC data, we selected 320 documents, 1064 queries, and 645,271 terms. The relatively small number of documents made it feasible to perform the multiple experiments described here. Ongoing research will address larger sets of documents. We applied the LSI algorithm to this collection and kept various percentage of the singular values: k = 10%, 20%, …, 80%. We then compared the ranking result of consecutive permutations of k using the following

k	10%	20%	30%	40%	50%	60%	70%	80%
10%								
20%	0.64 ± 0.21							
30%	0.50 ± 0.25	0.70 ± 0.17						
40%	0.42 ± 0.24	0.52 ± 0.28	0.67 ± 0.21					
50%	0.37 ± 0.22	0.42 ± 0.31	0.52 ± 0.31	0.75 ± 0.19				
60%	0.33 ± 0.23	0.33 ± 0.30	0.40 ± 0.35	0.59 ± 0.33	0.78 ± 0.22			
70%	0.29 ± 0.23	0.28 ± 0.29	0.33 ± 0.33	0.48 ± 0.39	0.66 ± 0.35	0.84 ± 0.18		
80%	0.27 ± 0.22	0.25 ± 0.29	0.29 ± 0.33	0.43 ± 0.42	0.59 ± 0.41	0.76 ± 0.27	0.89 ± 0.14	
90%	0.26 ± 0.21	0.23 ± 0.30	0.26 ± 0.33	0.39 ± 0.43	0.54 ± 0.46	0.70 ± 0.33	0.83 ± 0.27	0.92 ± 0.19

Table 1. Average sample probability ± 95% confidence intervals that the first 10% elements of the first ranking using k singular values (rows) are contained in the first 10% of the elements of the second ranking using k singular values (columns).

Sample Probability measurements: We argue that if LSI is stable, then the top 10% of each result needs to have almost the same elements as another ranking with different cutoff of singular values. So, we calculate the sample probability that the first 10% elements of the first ranking are contained in the first 10% of the elements of the second ranking. This was done for each query. We then took the average over all 1064 queries. Ideally, we want this sample probability to be as close to one as possible.

For example, for each query we run LSI with k = 10% of maximum k and with k = 20% of maximum k. We then obtain the ranking from the two values of k. We construct the sample probability that the raking from LSI 20% is contained in the ranking of LSI 10%. We then take the average of the resultant sample probabilities from all 1064 queries. We repeat for LSI 10% vs. LSI 30%, etc. The resultant average sample probability and the corresponding 95% (sample) confidence interval are summarized in Table 1.

We observe that the confidence intervals are very large, ranging between 0.14 and 0.46. We also observe that the sample probability is at its max when comparing performance of LSI k with LSI k + 10% of max k and decreases monotonically as the difference between the number of singular values incorporated in LSI increases. For example, this probability is 0.64, 0.50, 0.42, 0.37, 0.33, 0.29, 0.27, 0.26 for 10% vs. 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, respectively. On the other hand, the maximum average sample probability achieved by comparing LSI k with LSI k + r increases generally as k increases, reaching as high as 0.92 when comparing LSI 80% with LSI 90%. However, running LSI in this case takes almost a week on a fast machine, and it defeats the rationale behind the use of LSI vs. VSM.

Thus, it is clear from our analysis that LSI is not robust with respect to the choice of the number of singular values. In fact, the ranking results of LSI depend on the choice of k no matter how high the chosen number of singular values is. This alone brings the integrity of LSI results into question. Add to this the fact that LSI is orders of magnitude slower than VSM. Based on the evidence from this test, we must conclude that LSI may be insufficiently robust as a ranking algorithm for information retrieval.

6. Conclusions

In this paper, we have investigated the stability of the latent semantic indexing (LSI) algorithm. We applied it to sub-collections from TREC Million Query datasets using different singular value cut-offs. The results of our analysis show that the LSI ranking is very sensitive to the choice of the number of singular values retained. This put the robustness of LSI into question. Our ongoing research will include larger collections, including the 2009 TREC Web track dataset of over 1 billion documents. Further analysis of other sub-collections, and comparison of ranking stability to TREC evaluation measures such as mean average precision, are needed in order to better understand the impact of this variability in LSI rankings for different numbers of singular values.

References

[1] L. Calvé and J. Savoy, "Database merging strategy based on logistic regression." Information Processing and Management, 2000, 36(3):341-359.

[2] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," Journal of the Society for Information Science, 1990, 41(6):391-407.

[3] M. Efron, "Eigenvalue-based model selection during latent semantic indexing," Journal of the American Society for Information Science and Technology, 2005, 56(9):969-988.

[4] C. Fallen, G. B. Newby, and K. McCormick, "Distributed multisearch and resource selection for the TREC Million Query track." Gaithersburg: National Institute of Standards and Technology, 2008.

[5] K. Gamiel, G. B. Newby, and N. Nassar, "Grid information retrieval requirements (GFD-I.027)." Lamont, Illinois: Open Grid Forum, 2003.

[6] E. M. Garzón and I. García, "Solving eigenproblems on multicomputers: two different approaches," International Journal of Computers and Applications, 2004, 26(4):213-222.

[7] M. R. Guarracino, F. Perla, and P. Zanetti, "A parallel block Lanczos algorithm and its implementation for the evaluation of some eigenvalues of large sparse symmetric matrices on multicomputers," International Journal on Applied Mathematics and Computer Science, 2006, 16(2):241-249.

[8] S. Raghavan and H. Garcia-Molina, "Crawling the hidden web," Proceedings of Very Large Databases (VLDB), 2001.

[9] G. Salton, A. Wong, and C. S. Yang, "A Vector Space Model for Automatic Indexing," Communications of the ACM, 1975, 18(11):613-620.

[10] L. Schamber, M. B. Eisernberg, and M. Nilan, "A re-examination of relevance: toward a dynamic, situational definition," Information Processing and Management, 1990, 26(6):755-776.

[11] E. Voorhees, "Overview of TREC 2007," Proceedings of the 16th Annual Text REtrieval Conference. Gaithersburg: National Institute of Standards and Technology, 2007.

[12] J. Xu, H. Chen, Z. Zhou, and J. Qin, "On the topology of the dark web of terrorist groups," Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI), 2006.

[13] G. Golub and A. van Loan, "Matrix Computations," Johns Hopkins University Press, 1996.

[14] H. Kettani and G. B. Newby, "On the ranking of text documents from large corpuses," proceedings of the 2009 International Conference on Data Mining (DMIN'09), Las Vegas, Nevada, July 2009