Combining Latent Semantic Indexing and Clustering to Retrieve and Cluster Biomedical Information: A 2-step Approach

Jon Rune Paulsen and Heri Ramampiaro
Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU),
N-7491 Trondheim, Norway
heri@idi.ntnu.no

Abstract
This paper presents document retrieval approach based on combination of latent semantic index (LSI) and two different clustering algorithms. The idea is to first retrieve papers and create initial clusters based on LSI. Then, we use flat clustering method to further group similar documents in clusters. The paper also presents a new algorithm for k-means clustering that aims at dealing with the fact that the standard k-means algorithm is too greedy. Our experiments show that in many of cases the two-step algorithm performs better than standard k-means. The main advantage of our method is that it forces the centroid vector towards the extremities, and consequently gets a completely different starting point compared to the standard algorithm. This also makes the algorithm less greedy than the standard one. We believe our method can be used to retrieve relevant documents from a document collection. Our experiments have revealed that it performs well in most cases, but also failing in some cases.

1 Introduction and Motivation

Search engines are mostly developed and associated with retrieval of Internet web pages. However, other research communities such as the biology community have recognised the usefulness of search engines, and they have long been important tools for searching biomedical information. Since the 1960s, computers have been important tools in biology for storing and retrieval of (biomedical) information [9]. The biomedical field produces a lot of information and the amount is growing very fast, making the development of effective search tools crucial. This is true, despite the fact that there already exist a lot of search tools for biomedical information. Just like standard search engines, the amount of data still gives rise to issues for the developers and the users such as: How to store the data? How to reach the required information? How to retrieve information in a quick and accurate way? Are the retrieved data relevant to the users? Are the users satisfied with the search results?

This paper was presented at the NIK-2009 conference; see http://www.nik.no/.

As an attempt to address the above questions, in this paper we investigate the effects of combining Latent Semantic Indexing (LSI) and a flat clustering method for retrieval of biomedical information. The main contributions of the paper is the proposed *2-step k-means algorithm* as an improvement of the well-known k-means clustering algorithm. In addition, we present a new way to combine LSI and the k-means algorithms (i.e., both our new algorithm and the original one) to improve search and retrieval of biomedical information.

The rest of this paper is organised as follows. First to put our work in perspective, Section 2 discusses relation to other work. Then, in Section 3 we discuss our approach, explaining first the underlying ideas of LSI and its related algorithms, and second introducing our 2-step k-means algorithm. Thereafter, Section 3 discusses the implementation of the prototype, including the LSI and the clustering algorithm. Based on this implementation, in Section 4 we evaluate our approach, discussing its benefits and weaknesses. Finally, Section 5 summarises and concludes our paper.

2 Related Work

There are only a few works that we are able to identify as directly related to our work, i.e., combining LSI with clustering. This section presents three different proposals that we can compare our work with.

Gleich and Zhukov [8] evaluated the application of the singular value decomposition (SVD) to a search term suggestion system. They investigated the effect of SVD subspace projections for term suggestion ranking and clustering. Their study concentrated on the clustering behaviour when applying LSI. Although they do combine clustering and LSI, this work seems different from ours in that (1) they do not include cosine similarity measures, and (2) they do not apply k-means clustering in combination with LSI.

The second approach worth mentioning is the work of Ding [4]. He proposed a k-means algorithm called aspherical k-means. This method was introduced for clustering documents, where the cluster centroids are identified as concept vectors and compared to LSI index vectors. The main finding of this work is that the subspace covered by the concept vectors are close to the LSI subspace. Using this knowledge, the method has been further developed into concept indexing [4]. Although this method seems to be similar to our approach, the main difference from ours is that we use the concepts from the LSI to further group similar documents into clusters with the k-means clustering algorithms, rather than using the LSI concepts as clusters per se.

The third and final approach is by Jing et al. [14]. They present a text clustering system developed based on a k-means type subspace clustering algorithm to group high dimensional and sparse text data. They add a new step in the k-means clustering process to automatically calculate the feature weights for each cluster so that important features forming a cluster subspace can be identified by the weight values. This extension enables the k-means clustering algorithm to cluster high dimensional text data in subspaces of the keyword features, so that sparsity of text data can be effectively handled. The additional step does not increase the number of iterations, and the performance of the k-means clustering process is preserved. The introduction of a new step in the k-means algorithm is the only similarity of this work to ours. The usage of this extra step is however different from how our 2-step algorithm is designed. While the main usage of the extra step in this approach is only for term re-weighting, our extra step is aimed at making the k-means algorithm less greedy. Moreover, they do not apply LSI as a starting point before the clustering.

In summary, as far as we know, the work presented in this paper seem to be unique in terms (1) the usage of LSI to prepare for clustering and (2) the introduction of the new step in the k-means algorithm. How this works is discussed in the next section.

3 The Approach

In this section, we present our method by first giving an overview of Latent semantic indexing (LSI) including how we use it, and the flat clustering method k-means. Then, we will present our own extension of the standard k-means method.

The Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI) is an approach for automatic indexing and retrieval of documents based on the notion of *concept*. Founded on the vector space model (VSM), it was designed to overcome fundamental challenges such as retrieving documents based on the conceptual content, where individual words would provide unreliable evidence about the conceptual topic or meaning of a document.

LSI was designed to retrieve documents based on concept matching rather than index term matching. Matching based on concept allows documents to be retrieved even though they are not indexed by the query index terms. For this to be possible, an underlying latent semantic structur in the data must be present. And, by treating the unreliability of observed term-document association data as a statistical problem, LSI tries to overcome the deficiencies of term-matching retrieval. Then, to estimate the latent structure and remove the obscuring ”noise” from statistical techniques, LSI uses the well-known matrix method Singular Value Decomposition (SVD). Using SVD, a semantic space is constructed from a large matrix of term-document association data. The original matrix is replaced with a low rank approximation by using the SVD - the SVD creates three minor matrices, a left and a right singular vector matrix, and a diagonal matrix of singular values [13]. In the semantic space the terms and documents that are closely associated, are placed near each other [3, 15]. The semantic space contains a term-concept space and a document concept space; the left and right singular vector matrices, respectively.

In our work, we mainly use LSI in the initial retrieval of documents. In this way, we take advantage of the aforementioned concept, and use it as starting point for the clustering. How this is done is explained later.

As alternative to the original LSI, we also investigated two other variants. These are Probabilistic Latent Semantic Indexing (PLSI) [4] and Variable Latent Semantic Indexing (VLSI) [2] that we will due to the space constraints refer to the literature for more detailed information. We have chosen the standard LSI method for our solution, mainly because it fits very well with the vector space model that our approach is built on. Moreover, although both PLSI and VLSI both are interesting, there are contradictory reports about PLSI’ s performance, and VLSI is a new approach with only limited information available on its performance.

The 2-step k-means Clustering Algorithm

Since not all relevant items can be retrieved and the whole document space has to be searched, the retrieval effectiveness and efficiency are low. By grouping similar documents into cluster these disadvantages can be overcomed [15]. In the following two cluster methods will be described. Both of the methods are based on all pairwise document similarities and assembles similar items into common clusters. In the approach based on pairwise similarities, each document is represented as a document vector as in the vector space model. Thereafter, the similarity between each pair of documents is calculated.

Standard k-means Clustering

Since our algorithm is based on the standard k-means clustering algorithm, we will first give an overview of how this algorithm works.

The k-means clustering algorithm is known to be efficient in clustering large data sets. This clustering algorithm was developed by MacQueen [16], and is one of the simplest and the best known unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given set of data with a certain number of clusters specified a priori, where each cluster is defined by a centroid. For the k-means method to be useful, the centroids must be placed as far away from each other as possible [6]. The next step is to add points, in this case document vectors, to the clusters. For each document vector, a similarity between the document and each of the clusters' centroids is computed. The document is added to the cluster which the centroid is the most similar to. When all of the document vectors have been added to a cluster the first step is completed. The centroids are then recalculated, meaning a mean vector is created for each of the clusters, based on the document distribution from the previous step. With the new centroids, a new binding has to be done between them and the same data set, creating a loop. The loop is repeated until the centroids do not changes location any more, meaning the centroids no longer moves.

The original k-means algorithm is composed of the steps shown in Algorithm 1. This

Algorithm 1 The original k-means algorithm [11]

```
Require: The initial number of clusters $k$ {Place k points into the space represented by
  the objects that are being clustered. These points represent initial group centroids.}
  repeat
    for $j = 0$ to $n$ {I.e., for each document $d_j$ in the collection} do
      $min_{sim} := sim(d_j, c_0)$
      $t := 0$
      for $i = 1$ to $k$ {I.e., for each cluster $C_i$} do
        Calculate the similarity $sim(d_j, c_i)$ between a document $d_j$ and a centroid $c_i$
        if $sim(d_j, c_i) < min_{sim}$ then
          $min_{sim} := sim(d_j, c_i)$
          $t := i$
        end if
      end for
      $C_t \leftarrow d_j$ {Assign $d_j$ to the cluster $C_t$ with the closest centroid}
    end for
    for $i = 1$ to $k$ do
      Recalculate the positions of the centroid of $C_i$
    end for
  until the global centroids no longer move
  return $k$ clusters of documents
```

procedure will always (or mostly) terminate. However, because the algorithm is sensitive to the initial centroid selection, the algorithm does not necessarily find the most optimal configuration of clusters. A way to reduce this effect is to run the algorithm multiple times, which eventually could lead to an optimal solution.

Another issue with k-means is that it is a greedy algorithm. It partitions *n* samples into *k* clusters so as to minimize the sum of squared distances to the cluster centers. In addition, there are other weaknesses with this algorithm that is worth mentioning [11]:

- The initialization of the means is not specified. One popular way to start is to randomly choose k of the samples.
- The initial values for the means affects the result and suboptimal partitions are frequently found. A standard approach is to try a number of different starting points.
- It can happen that the set of samples closest to a particular cluster is empty, so that the cluster cannot be updated.
- The results depend on the metric used to measure the similarity between the samples and the clusters. A popular solution is to normalize each variable by its standard deviation, though this is not always desirable.
- The results depend on the value of k, meaning that the number must be set beforehand.

The last point is especially troublesome, because there are normally no easy way of knowing how many clusters exist. Determining the optimal number of clusters for any given data set is a difficult task. One approach is to compare results of multiple runs with different number of clusters and choose the one that give the best result according to a given criterion, although by increasing the k value may increase the risk of overfitting [11]. However, Fraley and Raftery [7] have proposed an analysis to determine both the best choice of number of clusters and a working method.

There are other variants of the k-means algorithm. The recent development of the new k-means type algorithms with variable weighting ability enables the efficient k-means clustering process to discover clusters from subspaces that are identified by the weights of variables. One method in this respect is the method proposed by Jing et al. [14] we discussed in Section 2.

The 2-step k-means Clustering

In order to deal with the greediness of the standard k-means algorithm we propose a 2-step k-means algorithm. This version aims at being less greedy than the standard version concerning the distribution of the documents. Unlike the standard version, the 2-step k-means consists of two modes, *a local* and *a global mode*. In the local mode, ÒlocalÓ centroids are calculated based on the distributed documents such that their similarity values are higher than a specific marginal value, which is, for our purpose, set to 0.8. A document is compared to more than one cluster centroid. Then, the document is assigned to a cluster only if the similarity value for the document and the cluster centroid exceeds the marginal value. In other words, if the similarity values of documents against all cluster centroids are below the marginal value, the document will not be added to any of the clusters. The main goal of this is to force the centroids away from each other, and thus giving the clustering algorithm a better foundation for clustering the documents.

The 2-step k-means algorithm is shown in Algorithm 2. As we can see, it is not significantly more complex than the standard k-means. As can be observed, we introduced one more step which could lead to an increased runtime and thus worsening the overall performance. However, the benefit of having a less greedy algorithm and thus a more optimal clustering is more important than the performance, in the long run.

Algorithm 2 The 2-step k-means algorithm

```
Require: The initial number of clusters $k$
  {*STEP I: Local mode*}
  repeat
    for $i = 0$ to $k$ {*I.e., for each cluster $C_i$*} do
      Calculate the similarity $sim(d_j, c_i)$ between a document $d_j$ and a centroid $c_i$
      if $\exists! t \mid sim(d_j, c_t) \geq r$ {*where r is a marginal value*} then
        $C_i \leftarrow d_j$ {Assign $d_j$ to the cluster $C_i$}
      end if
    end for
    for $i = 1$ to $k$ do
      Recalculate the positions of the *local* centroids $C_i$.
    end for
  until The local centroids stop moving.
  {*STEP II: Global mode*}
  repeat
    for $j = 0$ to $n$ {*I.e., for each document $d_j$*} do
      $max_{sim} := sim(d_j, c_0)$
      $t := 0$
      for $i = 1$ to $k$ {*I.e., for each cluster $C_i$*} do
        Calculate the similarity $sim(d_j, c_i)$ between a document $d_j$ and a centroid $c_i$
        if $sim(d_j, c_i) > max_{sim}$ then
          $max_{sim} := sim(d_j, c_i)$
          $t := i$
        end if
      end for
      $C_t \leftarrow d_j$ {Assign $d_j$ to the cluster $C_t$ with the closest centroid}
    end for
    for $i = 1$ to $k$ do
      Recalculate the positions of the centroid of $C_i$
    end for
  until the *global* centroids no longer move
  return $k$ clusters of documents
```

Nevertheless, there are other issues that we would like to point out. First, too low ratio, mainly because the initial centroids are too close from each other, may lead to clusters being empty. This could be a challenge as these clusters will not be able "get back in" again. For this reason, good initial selection of centroids is also crucial here. Further, when the local centroids have been found, the algorithm may start being greedy. The 2-step algorithm can still be seen an improvement as it will, in the long run, be less greedy than the standard one. In Section 4 an evaluation of the algorithm is presented.

Implementation

To be able to evaluate our approach, we have implemented a proof-of-the-concept prototype. Instead of re-inventing the wheel, the implementation of our prototype is based on use of existing open source libraries. Because of our goals related to performance, extensibility and simplicity as well as scalability, our choice has felt on Java Lucene [12].

In addition, we implemented LSI using Java JAMA API[1].

Figure 1 shows how the prototype is built. The system consists of two main modules: **preprocessing** of documents and the main **application**. The preprocessing part mainly consists of the document repository for raw documents and a parser that parse and split the raw documents into a set of documents that can be indexed by Lucene. The second part consists of two sub-modules; a Lucene indexing module and a searching and clustering module. The Lucene indexing module is used to index documents and parse queries. The main role of the searching and clustering module is to reprocess the lucene index to make it possible to do LSI/SVD, and clustering with both standard k-mean and 2-step k-means as described in Section 3 .

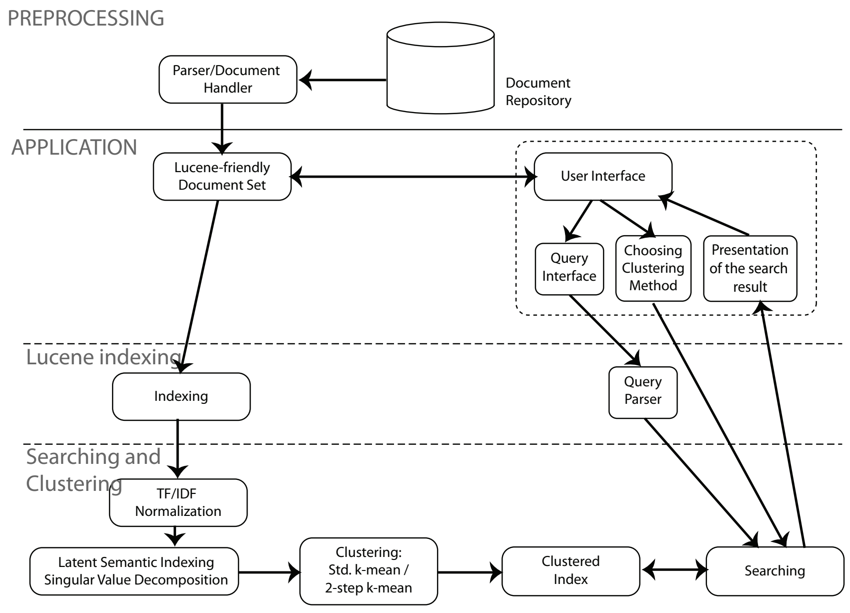

Figure 1: **System architecture**

As shown in Figure 1, both k-means algorithms were implemented. They were designed in to separate modules. The standard k-means module implements Algorithm 1. It reads the index statistics data, such as number of documents, number of index terms and the value of *k*, from a file. Based on these data, the reduced right singular value matrix is reconstructed, which also constitutes the documet-concept space as described by Deerwester [3]. Further, to determine the initial centroid vectors, each centroid gets the vector of a document. The initial allocation of the centroid vectors are mainly done by guessing. After the clusters are created, they are populated. Then, for each document vector, the similarity between the document and the centroid vectors is calculated. The document is added to the cluster containing the centroid vector with highest similarity. This is repeated until the centroid vectors no longer move. Finally, the clusters are written to files with a corresponding cluster list. This list contains information about the clusters and their centroid vectors.

The 2-step k-means module implements Algorithm 2. This means it works in the same fashion as described above. The main difference from the standard algorithm is readily the introduction of a local and a global modes, where local and global centroid vectors are added to the clusters. A restriction on the allocation of documents to clusters is also introduced in this method. As mentioned earlier, to be added to a cluster in Òlocal modeÓ, a document must not fit in more than one cluster, and it must not exceed the

[1]See http://math.nist.gov/javanumerics/jama/

marginal value. Further, in local mode, the clustering method sorts out documents that have lower similarity than the given marginal value. The intended outcome of this is driving the clusters away from each other.

When the centroids no longer move in local mode, it switches to global mode. The global mode works just like standard k-means but the initial centroid vectors are different because they have been driven away from each other already.

4 Achievements and Limitations

The developed search engine application provides indexing and searching operations for biological information stored in the Online Mendelian Inheritance in Man (OMIM) collection [10]. In this section, we will evaluate our approach with respect to how the method/application works (acceptance tests), performance, and retrieval accuracy.

Results from the Acceptance Tests

To evaluate our method, we conducted our experiments using the OMIM database. The main goal of the tests is to investigate the impact of index clustering on the actual search results, and to compare the behaviours of the two k-means methods. The standard k-mean method was set as the reference method, meaning that our tests were mainly done to compare how our new algorithm behave compared to the standard algorithm. All tests of the implemented application are executed with a marginal value of 0.80.

For each cluster the 2-step k-means optimally yield a more fairly distribution of the documents and discover a more optimal centroid vector, when based on the same initial basis, than the standard k-means do. Which one of the two clustering methods performs best depends on which trade-offs one emphasises, i.e., whether speed is more important than accuracy. We recognise that the system has its limitations in terms of search speed. However, when executed on a clustered LSI index, the search time is reduced to approximately a third of the time compared to a non-clustered index. This reduction in runtime comes from the reduction in search space - i.e. the number of dimensions is reduced considerably. Moreover, the clusters created by the 2-step method has a more uniform distribution than the clusters created by the standard k-means. This is because the centroid vectors are forced towards the extreme points. As Faraoun and Boukelif [6] mentions in their article, to utilize the k-means method, the best way is to place the initial centroid vectors as far away from each other as possible. In light of this the 2-step k-means method may be viewed as an semiautomatic method for forcing the centroid vectors as far as possible away from each other. To this extent the goal of being less greedy has been met, especially when the algorithm runs in the local mode. However, when turning to global mode it works similar to the standard k-means again, and thus it could be greedy. Nevertheless, with the 2-step k-means the result sets are in some cases more concentrated and better than with the standard k-means, while in other cases the results are not satisfactory, mainly due to the aforementioned greediness.

By doing conceptual search, one would like to retrieve documents that does not contain the query terms. This has its advantages for conceptual queries. Consider, for example, the query: ”*proteinase inhibitors serpin*”. This retrieves several documents containing the term ”*protein*” although this term does not have any direct lexicographic/grammatical relation to ”*proteinase*”. ”*Proteinase*” is stemmed to ”*proteinas*”. So, applying exact match methods like boolean queries, one would not be able to retrieve documents containing the term ”*protein*”. By using LSI/SVD, however, we were still able to retrieve these documents. This because ”proteinase” and ”protein” are conceptually similar according to the structure created by the SVD reduction.

Accuracy

The lack of test collection focusing on the OMIM database has made difficult to conduct precise evaluation. Moreover, since the LSI method matches concepts instead of exact matching, the precision of the application may be challenging to verify. We have, nevertheless conducted an evaluation on the accuracy of the clustering processes and the search results. The LSI method works satisfactory. It ranks the relevant documents first for 90 % of our test cases, which means an average precision of 90%.

When performing a search with the combination of LSI and clustering, the accuracy fluctuates. This may be caused by the distribution of the documents, and by the following factors. First, treating a document as a whole during the indexing and clustering processes without taking any consideration of the various fields (TI, TX, CD, etc.). A solution is to index the documents by taking into account the fact that some fields are more important than others, and therefore have to be indexed differently. The possible benefit is increased accuracy. Second, the rank k value may be too high or low, and therefore fails to remove the neccessary noise. The value of the rank k influences the latent structure and the removal of noise significantly. According to the literature [1, 3, 5], for hundred thousands of documents a rank k value of 100 seems to give the best results. Third, the clustering methods creates constant boundaries between the clusters and therefore split up, for instance two documents that are conceptually similar into two different clusters. Fourth, the use of substring matches during search generates a query vector with noise.

Performance

Indexing process: The Lucene indexing module utilizes the compound file index structure. Multifile index creates multiple files stored on separate segments of the index. By contrast, compound index consists of three files; deletetable segments, and a file containing the indexed documents and their field values. The multifile index structure uses shorter time to index the documents than the compound file index. The reason for this is that the compound index structure merges all of the files into one single file. To make use of the Lucene index in different stages the index is stored in a file system. Now, to increase the indexing speed we could use a memory-based indexing based on Lucene RAMDirectory instead of the more standard file system-based indexing, i.e., the Lucene FSDirectory. However, we decided to use the FSDirectory index for two main reasons. First it helps us to do the indexing in several independent steps, which is required because of both the latent semantic indexing and the clustering processes. Second, in the long run, the indexing part is mostly a static process, and therefore the indexing performance is not so crucial.

Clustering process: What we were particularly interested to find out was the how our 2-step k-means algorithm performed compared to the standard one. Our experiments have shown that there are some differences in the clustering process speeds. The standard k-means clustering method had execution times within the interval of 0,5 - 1,5 minutes, depending on the initial centroid vectors. The average execution time was measured to 1 minute.

As expected due to the extra step, the 2-step k-means clustering method had an average running time twice as much as the standard one. Its runtime interval was 1 - 2,5 minutes.

5 Conclusion and Future Work

This paper has presented an approach combining clustering algorithm with latent semantic indexing (LSI) to retrieve biomedical information. The clustering algorithms that we used in our approach is based on the k-means clustering algorithm. As discussed in this paper, the standard k-mean algorithm can be seen as a greedy algorithm. We therefore proposed a new 2-step k-means algorithm aiming to improve the standard algorithm.

The tests have shown that in many of cases the 2-step algorithm performs better than the standard k-means. The main advantage of our method is that it force the centroid vector towards the extremities, and consequently gets a completely different starting point as compared to the standard k-means method. This also makes the method less greedy than the standard one, which in itself is an improvement.

Despite the above improvements, there are still some issues that we need to address. First, although the staring point is completely different from the one with the standard k-mean method, the global step is similar to the standard one, and thus the effect on the final result is not big. Second, if a cluster does not get any documents, it will remain empty when the clustering process is finished, which is not good. However, if this happens, it is straightforward to correct it by letting a cluster to keep its old centroid vector. A way to avoid a cluster being empty is to limit the number of documents to be clustered in each iteration and set an optimal value for the marginal value in the local mode. As discussed, this would determine how far from each other the centroids should be. Another side effect of retaining the old centroid vector is to make the 2-step method less sensitive to the initial location of the initial centroid vectors.

Regardless of the above challenges, we believe our method can be used to retrieve relevant documents from a document collection. Our experiments have revealed that it performs well in most of the cases, although failing in some cases, and there are still room for improvements. Nevertheless, our future work will focus on improving the sensitivity of the location of the initial centroids, and dealing with the greediness in global modes.

Acknowledgements

Parts of this paper was written while visiting the Tsujii Lab at the University of Tokyo. We appreciate their hospitality and for providing a good research environment. We would also thank Jon Olav Hauglid for his comments and feedback.

References

[1] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM(4)*, 37(4):573–595, 1995.

[2] A. Dasgupta, R. Kumar, P. Raghavan, and A. Tomkins. Variable latent semantic indexing. In *ACM SIGKDD 2005*, 2005.

[3] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[4] C. H. Q. Ding. A probabilistic model for latent semantic indexing. *Journal of the American Society for Information Science and Technology*, 56(6):597–608, 2005.

[5] Miles Efron. Eigenvalue-based model selection during latent semantic indexing: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 56(9):969–988, 2005.

[6] K. M. Faraoun and A Boukelif. Neural networks learning improvement using the kmeans clustering algorithm to detect neural intrusions. *IJCI*, 3(2):161–168, 2006.

[7] C. Fraley and A. E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *Computer Journal*, 41(8):578–588, 1998.

[8] David Gleich. Svd subspace projections for term suggestion ranking and clustering. In *In Technical Report, Yahoo! Research Labs*, 2004.

[9] J. B. Hagen. The origin of bioinformatics. *Nature Reviews: Genetics*, (1):231–236, 2000.

[10] Ada Hamosh, Alan F. Scott, Joanna Amberger, Carol Bocchini, David Valle, and Victor A. McKusick. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research*, 30(1):52–55, 2002.

[11] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Kluwer Academic Publisher, 2000.

[12] Erik Hatcher and Otis Gospodnetic. *Lucene in Action*. Manning Publications Co., 209 Bruce Park Ave., Greenwich, CT 06830, 2005.

[13] B Hendrickson. Latent semantic analysis and fiedler embeddings. In *Proceedings of SIAM Workshop on Text Mining*, 2006.

[14] L. Jing, M. K. Ng, X. Yang, and J. Z. Huang. A text clustering system based on kmeans type subspace clustering and ontology. *International Journal of Intelligent Technology*, 1(2):91–103, 2006.

[15] G. Lu. *Multimedia Database Management Systems*. Artech House, 1999.

[16] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkley Symposium on Mathematical Statistical and Probability*, pages 281–297. University of California Press, 1967.