2.1 Introduction to Vector-Space Models

The vector-space models for information retrieval are just one subclass of retrieval techniques that have been studied in recent years. The taxonomy provided in [4] labels the class of techniques that resemble vector-space models ``formal, feature-based, individual, partial match'' retrieval techniques since they typically rely on an underlying, formal mathematical model for retrieval, model the documents as sets of terms that can be individually weighted and manipulated, perform queries by comparing the representation of the query to the representation of each document in the space, and can retrieve documents that don't necessarily contain one of the search terms. Although the vector-space techniques share common characteristics with other techniques in the information retrieval hierarchy, they all share a core set of similarities that justify their own class.

Vector-space models rely on the premise that the meaning of a document can be derived from the document's constituent terms. They represent documents as vectors of terms $d = (t_1, t_2, \ldots, t_n)$ where $t_i (1 \le i \le n)$ is a non-negative value denoting the single or multiple occurrences of term **i** in document **d**. Thus, each unique term in the document collection corresponds to a dimension in the space. Similarly, a query is represented as a vector $q = (\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_m)$ where term $\hat{t}_i (1 \le i \le m)$ is a non-negative value denoting the number of occurrences of $\hat{t}_i$ (or, merely a **1** to signify the occurrence of term $\hat{t}_i$) in the query [4]. Both the document vectors and the query vector provide the locations of the objects in the term-document space. By computing the distance between the query and other objects in the space, objects with similar semantic content to the query presumably will be retrieved.

Vector-space models that don't attempt to collapse the dimensions of the space treat each term independently, essentially mimicking an inverted index [11]. However, vector-space models are more flexible than inverted indices since each term can be individually weighted, allowing that term to become more or less important within a document or the entire document collection as a whole. Also, by applying different similarity measures to compare queries to terms and documents, properties of the document collection can be emphasized or deemphasized. For example, the dot product (or, inner product) similarity measure finds the Euclidean distance between the query and a term or document in the space. The cosine similarity measure, on the other hand, by computing the angle between the query and a term or document rather than the distance, deemphasizes the lengths of the vectors. In some cases, the directions of the vectors are a more reliable indication of the semantic similarities of the objects than the distance between the objects in the term-document space [11].

Vector-space models were developed to eliminate many of the problems associated with exact, lexical matching techniques. In particular, since words often have multiple meanings (polysemy), it is difficult for a lexical matching technique to differentiate between two documents that share a given word, but use it differently, without understanding the context in which the word was used. Also, since there are many ways to describe a given concept (synonomy), related documents may not use the same terminology to describe their shared concepts. A query using the terminology of one document will not retrieve the other related documents. In the worst case, a query using terminology different than that used by related documents in the collection may not retrieve any documents using lexical matching, even though the collection contains related documents [5].

Vector-space models, by placing terms, documents, and queries in a term-document space and computing similarities between the queries and the terms or documents, allow the results of a query to be ranked according to the similarity measure used. Unlike lexical matching techniques that provide no ranking or a very crude ranking scheme (for example, ranking one document before another document because it contains more occurrences of the search terms), the vector-space models, by basing their rankings on the Euclidean distance or the angle measure between the query and terms or documents in the space, are able to automatically guide the user to documents that might be more conceptually similar and of greater use than other documents. Also, by representing terms and documents in the same space, vector-space models often provide an elegant method of implementing relevance feedback [21]. Relevance feedback, by allowing documents as well as terms to form the query, and using the terms in those documents to supplement the query, increases the length and precision of the query, helping the user to more accurately specify what he or she desires from the search.

Information retrieval models typically express the retrieval performance of the system in terms of two quantities: precision and recall. Precision is the ratio of the number of relevant documents retrieved by the system to the total number of documents retrieved. Recall is the ratio of the number of relevant documents retrieved for a query to the number of documents relevant to that query in the entire document collection. Both precision and recall are expressed as values between **0** and **1**. An optimal retrieval system would provide precision and recall values of **1**, although precision tends to decrease with greater recall in real-world systems [11].

2.2 Latent Semantic Indexing

The Latent Semantic Indexing information retrieval model builds upon the prior research in information retrieval and, using the singular value decomposition (SVD) [14] to reduce the dimensions of the term-document space, attempts to solve the synonomy and polysemy (Section 2.1) problems that plague automatic information retrieval systems. LSI explicitly represents terms and documents in a rich, high-dimensional space, allowing the underlying (``latent''), semantic relationships between terms and documents to be exploited during searching.

LSI relies on the constituent terms of a document to suggest the document's semantic content. However, the LSI model views the terms in a document as somewhat unreliable indicators of the concepts contained in the document. It assumes that the variability of word choice partially obscures the semantic structure of the document. By reducing the dimensionality of the term-document space, the underlying, semantic relationships between documents are revealed, and much of the ``noise'' (differences in word usage, terms that do not help distinguish documents, etc.) is eliminated. LSI statistically analyses the patterns of word usage across the entire document collection, placing documents with similar word usage patterns near each other in the term-document space, and allowing semantically-related documents to be near each other even though they may not share terms [8].

LSI differs from previous attempts at using reduced-space models for information retrieval in several ways. Most notably, LSI represents documents in a high-dimensional space. Koll [16] for instance, used only seven dimensions to represent his semantic space. Secondly, both terms and documents are explicitly represented in the same space. Thirdly, unlike Borko and Bernick [6], no attempt is made to interpret the meaning of each dimension. Each dimension is merely assumed to represent one or more semantic relationships in the term-document space. Finally, because of limits imposed mostly by the computational demands of vector-space approaches to information retrieval, previous attempts focused on relatively small document collections. LSI is able to represent and manipulate large data sets, making it viable for real-world applications [7].

Compared to other information retrieval techniques, LSI performs surprisingly well. In one test, Dumais [8] found LSI provided 30% more related documents than standard word-based retrieval techniques when searching the standard MED collection. Over five standard document collections, the same study indicated LSI performed an average of 20% better than lexical retrieval techniques. In addition, LSI is fully automatic and easy to use, requiring no complex expressions or syntax to represent the query. Because terms and documents are explicitly represented in the space, relevance feedback [21] can be seamlessly integrated with the LSI model, providing even better overall retrieval performance.

The following sections briefly describe the LSI model. A computational example of the model can be found in [5].

2.2.1 Term-Document Representation

In the LSI model, terms and documents are represented by an $m \times n$ incidence matrix **A**. Each of the **m** unique terms in the document collection are assigned a row in the matrix, while each of the **n** documents in the collection are assigned a column in the matrix. A non-zero element a_{ij}, where

$$A = [a_{ij}],$$

indicates not only that term **i** occurs in document **j**, but also the number of times the term appears in that document. Since the number of terms in a given document is typically far less than the number of terms in the entire document collection, **A** is usually very sparse [5].

.2.2 Weighting

LSI typically uses both a local and global weighting scheme to increase or decrease the relative importance of terms within documents and across the entire document collection, respectively. The product of the local and global weighting functions is applied to each non-zero element of **A**,

$$a_{ij} = L(i, j) * G(i),$$

where $L(i,j)$ is the local weighting function for term **i** in document **j** and $G(i)$ is the global weighting function for term **i**. Among the popular local and global weighting functions tried with LSI (see [8], p. 233), Dumais found the log-entropy weighting scheme provided a 40% advantage over raw term frequency on several standard document test collections [8].

2.2.3 Computing the SVD

Once the $m \times n$ matrix **A** has been created and properly weighted, a rank-**k** approximation ($k \ll \min(m,n)$) to **A**, A_k, is computed using an orthogonal decomposition known as the singular value decomposition (SVD) [14]. The SVD of the matrix **A** is defined as the product of three matrices,

$$A = U \Sigma V^T,$$

where the columns of **U** and **V** are the left and right singular vectors, respectively, corresponding to the monotonically decreasing (in value) diagonal elements of Σ which are called the singular values of the matrix **A**. As illustrated in Figure 1, the first **k** columns of the **U** and **V** matrices and the first (largest) **k** singular values of **A** are used to construct a rank-**k** approximation to **A** via $A_k = U_k \Sigma_k V_k^T$. The columns of **U** and **V** are orthogonal, such that $U^T U = V^T V = I_r$, where **r** is the rank of the matrix **A**. A theorem due to Eckart and Young [15] suggests that A_k, constructed from the **k**-largest singular triplets of **A** (a singular value and its corresponding left and right singular vectors are referred to as a *singular triplet*), is the closest rank-**k** approximation (in the least squares sense) to **A** [5].

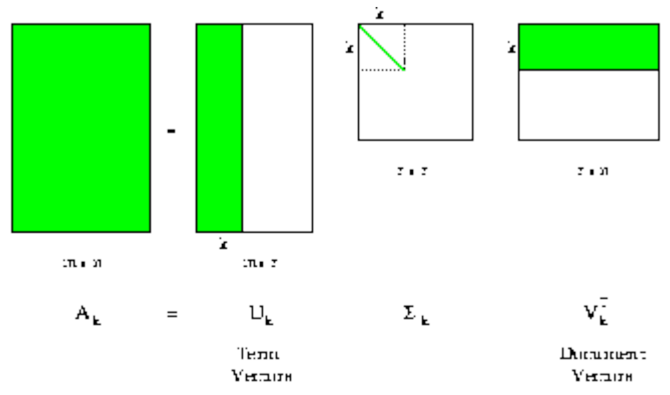

Figure 1: A pictorial representation of the SVD. The shaded areas of U and V, as well as the diagonal line in Σ, represent A_k, the reduced representation of the original term-document matrix A.

With regard to LSI, A_k is the closest **k**-dimensional approximation to the original term-document space represented by the incidence matrix **A**. As stated previously, by reducing the dimensionality of **A**, much of the ``noise'' that causes poor retrieval performance is thought to be eliminated. Thus, although a high-dimensional representation appears to be required for good retrieval performance, care must be taken to not reconstruct **A**. If **A** is nearly reconstructed, the noise caused by variability of word choice and terms that span or nearly span the document collection won't be eliminated, resulting in poor retrieval performance [5].

In LSI, the left and right singular vectors specify the locations of the terms and documents, respectively, in the reduced term-document space. The singular values are often used to scale the term and document vectors, allowing clusters of terms and documents to be more readily identified. Within the reduced space, semantically-related terms and documents presumably lie near each other since the SVD attempts to derive the underlying, semantic structure of the term-document space [5].

2.2.4 Query Projection and Matching

In the LSI model, queries are formed into *pseudo-documents* that specify the location of the query in the reduced term-document space. Given **q**, a vector whose non-zero elements contain the weighted (using the same local and global weighting schemes applied to the document collection being searched; see Section 2.2.2) term-frequency counts of the terms that appear in the query, the pseudo-document, $\hat{q}$, can be represented by

$$\hat{q} = q^T U_k \Sigma_k^{-1}.$$

Thus, the pseudo-document consists of the sum of the term vectors ($q^T U_k$) corresponding to the terms specified in the query scaled by the inverse of the singular values (Σ_k^{-1}). The singular values are used to individually weight each dimension of the term-document space [5].

Once the query is projected into the term-document space, one of several similarity measures can be applied to compare the position of the pseudo-document to the positions of the terms or documents in the reduced term-document space. One popular similarity measure, the cosine similarity measure, is often used because, by only finding the angle between the pseudo-document and the terms or documents in the reduced space, the lengths of the documents, which can affect the distance between the pseudo-document and the documents in the space, are normalized. Once the similarities between the pseudo-document and all the terms and documents in the space have been computed, the terms or documents are ranked according to the results of the similarity measure, and the highest-ranking terms or documents, or all the terms and documents exceeding some threshold value, are returned to the user [5].

2.2.5 Relevance Feedback

Relevance feedback is an effective method for iteratively improving a query without increasing the computational requirements to perform the query. Relevance feedback uses the terms contained in relevant documents to supplement and enrich the user's initial keyword query, allowing greater retrieval performance (in terms of precision and recall; see Section 2.1). Since LSI explicitly represents both terms and documents in the same space, a relevance feedback query is constructed in essentially the same way as a regular query.

Given two vectors:

q	from the previous section, specifying the terms in the query, and
d	a vector whose elements specify the documents in the query,

the pseudo-document representing the relevance feedback query is given by

$$\hat{q} = q^T U_k \Sigma_k^{-1} + d^T V_k.$$

Then, by matching the pseudo-document against each term or document vector (as described in the previous section) and sorting the results, the highest-ranking terms or documents (or all those that exceed some threshold value) can be returned to the user.