case (where all matching documents are returned), here the distance measure produces a ranking of all documents that include at least one relevant term.

14.3.3 Latent Semantic Indexing

In the schemes we have discussed so far for text retrieval, we have relied exclusively on the notion of representing a document as a T-dimensional vector of term weights. A criticism of the term-based approach is that users may pose queries using different terminology than the terms used to index a document. For example, from a term similarity viewpoint the term `data mining` has nothing directly in common with the term `knowledge discovery`. However, *semantically*, these two terms have much in common and (presumably) if we posed a query with one of these terms, we would consider documents containing the other to be relevant. One solution to this problem is to use a knowledge base (a thesaurus or ontology) that is created a priori with the purpose of linking semantically related terms together. However, such knowledge bases are inherently subjective in that they depend on a particular viewpoint of how terms are related to semantic content.

An interesting, and useful, alternative methodology goes by the name of *latent semantic indexing* (LSI). The name suggests that hidden semantic structure is extracted from text rather than just term occurrences. What LSI actually does is to approximate the original T-dimensional term space by the first k principal component directions in this space, using the $N \times T$ document-term matrix to estimate the directions. As discussed in chapter 3, the first k principal component directions provide the best set of k orthogonal basis vectors in terms of explaining the most variance in the data matrix. The principal components approach will exploit redundancy in the terms, if it exists. Frequently in practice there is such redundancy. For example, terms such as `database, SQL, indexing, query optimization` can be expected to exhibit redundancy in the sense that many database-related documents may contain all four of these terms together. The intuition behind principal components is that a single vector consisting of a weighted combination of the original terms may be able to approximate quite closely the effect of a much larger set of terms. Thus the original document-term matrix of size $N \times T$ can be replaced by a matrix of size $N \times k$, where k may be much smaller than T with little loss in information. From a text retrieval perspective, for fixed recall, LSI can increase precision compared to the vector-space methods discussed earlier.

An interesting aspect of the the principal component representation for the

document-term matrix is that it captures relationships among terms by creating new terms that may more closely reflect the semantic content of the document. For example, if the terms `database`, `SQL`, `indexing`, `query optimization` are effectively combined into a single principal component term, we can think of this new term as defining whether the content of a document is about database concepts. Thus, for example, if the query is posed using the term `SQL`, but the database-related documents in the set of documents contain only the term `indexing`, that set of database documents will nonetheless be returned by the LSI method (but would not be returned by a strictly term-based approach).

We can calculate a singular-value decomposition (SVD) for the document-term matrix $\mathbf{M}$ in table 14.2. That is, we find a decomposition $\mathbf{M} = \mathbf{USV}^T$. Here $\mathbf{U}$ is a 10×6 matrix where each row is a vector of weights for a particular document, $\mathbf{S}$ is a 6×6 diagonal matrix of eigenvalues for each principal component direction, and the columns of the 6×6 matrix $\mathbf{V}^T$ provide a new orthogonal basis for the data, often referred to as the principal component directions.

The $\mathbf{S}$ matrix for $\mathbf{M}$ has diagonal elements

$$\lambda_1, \ldots, \lambda_6 = \{77.4, 69.5, 22.9, 13.5, 12.1, 4.8\}.$$

In agreement with our intuition, most of the variance in the data is captured by the first two principal components. In fact, if we were to retain only these two principal components (as two surrogate terms instead of the six original terms), the fraction of variance that our two-dimensional representation retains is $(\lambda_1^2 + \lambda_2^2)/\sum_{i=1}^{6} \lambda_i^2 = 0.925$; i.e., only 7.5% of the information has been lost (in a mean-square sense). If we represent the documents in the new two-dimensional principal component space, the coefficients for each document correspond to the first two columns of the $\mathbf{U}$ matrix:

```
d1    30.8998   -11.4912
d2    30.3131   -10.7801
d3    18.0007    -7.7138
d4     8.3765    -3.5611
d5    52.7057   -20.6051
d6    14.2118    21.8263
d7    10.8052    21.9140
d8    11.5080    28.0101
d9     9.5259    17.7666
d10   19.9219    45.0751
```

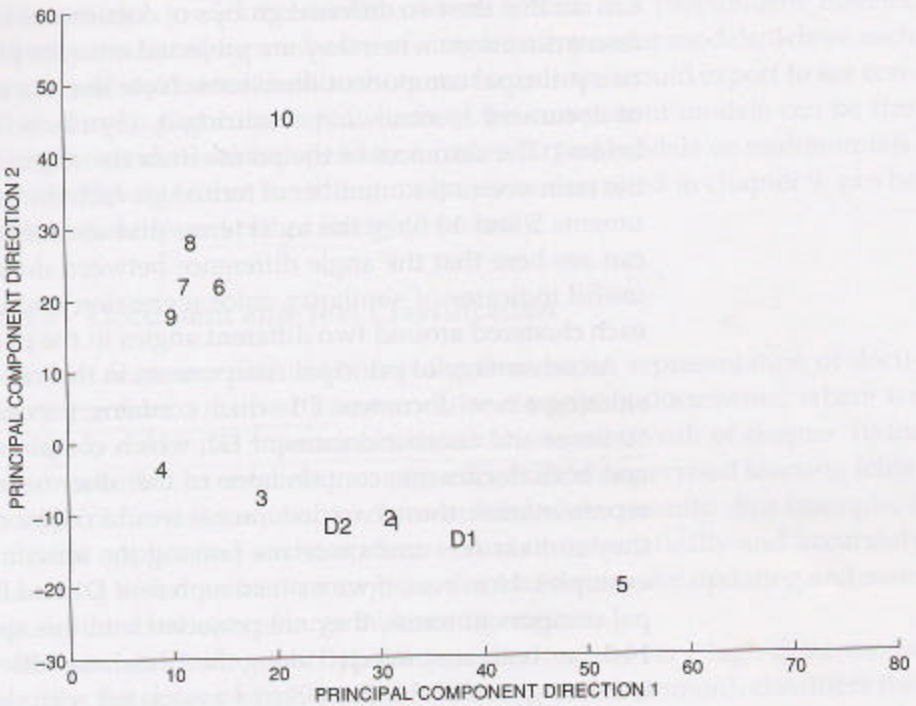

Figure 14.3 Projected locations of the 10 documents (from table 14.2) in the two dimensional plane spanned by the first two principal components of the document-term matrix M.

and we can consider these two columns as representing new *pseudo-terms* that are in effect linear combinations of the original six terms.

It is informative to look at the first two principal component directions:

$$\mathbf{v}_1 = (0.74, 0.49.0.27, 0.28, 0.18, 0.19) \tag{14.3}$$

$$\mathbf{v}_2 = (-0.28, -0.24, -0.12, 0.74, 0.37, 0.31). \tag{14.4}$$

These are the two directions (a plane) in the original six-dimensional term space where the data is most "spread out" (has the most variance). The first direction emphasizes the first two terms (*query, SQL*) more than the others: this is in effect the direction that characterizes documents relating to databases. The second direction emphasizes the last three terms *regression, likelihood, linear*, and can be considered the direction that characterizes documents relating to regression. Figure 14.3 demonstrates this graphically. We

can see that the two different groups of documents are distributed in two different directions when they are projected onto the plane spanned by the first two principal component directions. Note that document 2 is almost on top of document 1, somewhat obscuring it. (Symbols D1 and D2 are discussed below.) The distances of the points from the origin reflect the magnitude of the term-vector (i.e., number of terms) for each document. For example, documents 5 and 10 have the most terms and are furthest from the origin. We can see here that the angle difference between documents is clearly a very useful indicator of similarity, since regression and database documents are each clustered around two different angles in the plane.

An advantage of principal components in this context can be seen by considering a new document D1 which contains, for example, the term *database* 50 times and another document D2, which contains the term *SQL* 50 times, and both documents contain none of the other terms. In a direct keyword representation, these two documents would not be considered similar since they contain no common terms (among the specific terms used in our toy example). However, if we instead represent D1 and D2 using the two principal component terms, they are projected into this space, as shown in figure 14.3, i.e., both are projected along the "database" direction even though each is missing two of the three terms associated with databases. The principal component approach has implicitly modeled the interrelationship between terms. This feature can be taken advantage of for querying. Imagine now that we pose a query using only the term *SQL*. If we use the principal component representation (e.g., the first two pseudo-terms) for our documents, we can also convert the query into the pseudo-term representation, where again the query will be closer (in angle) to the database documents than the regression documents, allowing retrieval of documents that do not contain the term *SQL* at all but are related in content.

From a computational viewpoint, directly computing the principal component vectors (by seeking the eigenvectors of the correlation or covariance matrix for example) is usually neither computationally feasible or numerically stable. In practice, the PCA vectors can be estimated using special-purpose sparse SVD techniques that are well-suited for high-dimensional sparse matrices.

There are many other variations of this basic framework outline. The application of principal components for text retrieval is often referred to as latent semantic indexing (LSI). Use of this general technique has been shown in certain cases to improve retrieval performance in systematic tests, based on its ability to match documents and queries with no terms in common.