Chapter 2

2 Information Retrieval

In this chapter we review relevant work done in information retrieval. In particular, we introduce the Vector Space Model (VSM), one of the most widely used representations for documents. We then introduce and discuss several variations and extensions to basic VSM. We also describe several techniques for improving performance and metrics for measuring these improvements. Finally, we describe several tasks in information retrieval and identify those that most interest us in this work.

This review is not meant to be comprehensive; however, we believe that this chapter provides the background necessary to understand the contribution of this work to information retrieval. The reader who wishes either a deeper or broader discussion is referred first to [Frakes and Baeza-Yates, 1992]. Finally, this chapter assumes easy familiarity with linear algebra and eigenproblems. The reader for whom this is unfamiliar should feel free to refer to Appendix A for a discussion of the relevant concepts.

2.1 Issues in Information Retrieval

In Information Retrieval (IR) our basic task is to find the subset of a collection of elements that is relevant to a query. In Text Retrieval—the focus of this work—a query is an ordered set of English words and a collection is a set of natural language English documents.

Any text retrieval system must overcome the fundamental difficulty that the presence or absence of a word is insufficient to determine relevance. This is due to two intrinsic problems of natural language: *synonymy* and *polysemy*. Synonymy refers to the fact that a single underlying concept or idea can be represented by many different terms or combinations of terms (e.g. “car” and “automobile” often refer to the same class of objects). Polysemy refers to the fact that a single term can refer to more than one underlying concept or idea (e.g. “car” may be an automobile or the head of a LISP cons cell). Because of synonymy, it is difficult to realize that two documents describe the same topic when they use different vocabulary, leading to relevant documents being rejected (false negatives). Because of polysemy, it is difficult to realize that two documents that use some of the same terms describe different topics, leading to the retrieval of unwanted documents (false positives).

A variety of approaches have been developed to attack IR tasks in the face of these problems. We will focus on the popular Vector Space Model [Salton, 1971] representation for documents and queries. We will also focus on variations of latent semantic indexing [Deerwester *et al*, 1990], one technique designed to address synonymy and polysemy in the VSM framework and similar in flavor to the approach that we will derive.

2.2 The Vector Space Model

In the Vector Space Model (VSM), a document is a vector (see FIGURE 2-1). Each dimension represents a count of occurrences for a different word [Salton, 1971]. Queries are similarly represented, making queries no different from documents. A *collection* of documents is a matrix, **D**, where each column is a document vector $\mathbf{d_i}$. Thus, $\mathbf{D_{ij}}$ is the weight of word i in document j. Classically, the *similarity* between a document and a query, **q**, is defined to be the inner product of their vectors, $\mathbf{d^Tq}$. This approach may seem bizarre; however, the inner product is just a weighted match between the overlapping terms of two documents. Although expressed as linear algebra, it is essentially the same approach used by many search engines,

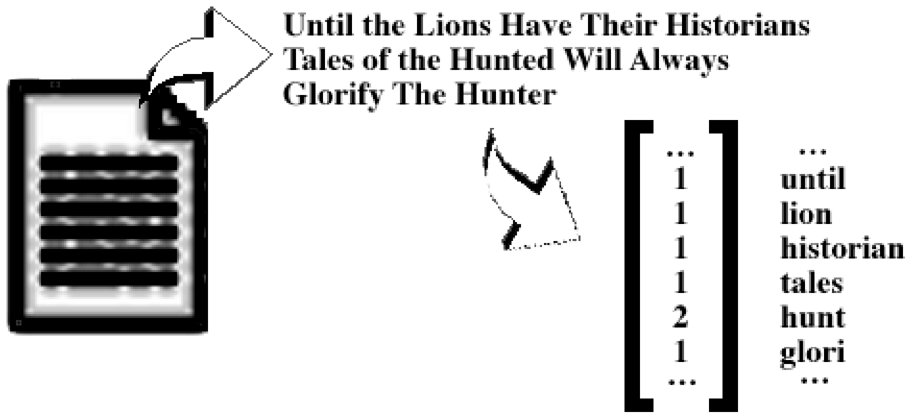
Until the Lions Have Their Historians
Tales of the Hunted Will Always
Glorify The Hunter
...
1
1
1
1
2
1
...
...
until
lion
historian
tales
hunt
glori
...

FIGURE 2-1. **The Vector Space Model.**
In the Vector Space Model, documents and queries are transformed into histograms of word counts. A collection of documents is a matrix.

from the library systems commonly available in universities to the wildly popular Alta Vista web search engine.

There are several advantages to this approach beyond its mathematical simplicity. Above all, it is computationally efficient to compute a histogram and requires very little space to store it. Notice that although document vectors live in a very high-dimensional space, the document matrix will be sparsely populated, made up mostly of zeroes. This is true because in general most documents will not contain most of the possible words. Thus algorithms for manipulating the matrix only require space and time proportional to the average number of different words that appear in a document, a number likely to be much smaller than the full dimensionality of the document matrix. Similarly, comparing a query to all the documents in a collection is efficient (in practice, it is done with an inverted term index). These are key advantages when collections may require gigabytes to store.

2.2.1 Similarity Measures and Term Reweighting

As we have noted before, the similarity between a VSM document $\mathbf{d}$, and a VSM query $\mathbf{q}$, is defined to be their inner product, $\mathbf{d}^T\mathbf{q}$. Because documents in a collection may be of varying

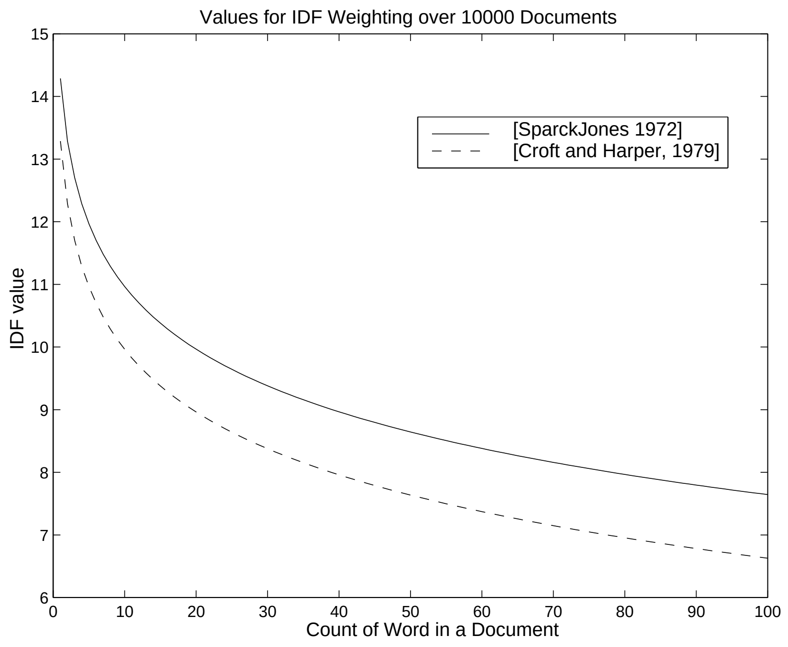

FIGURE 2-2. **Two Variations of Inverse Document Frequency.**
IDF re-weights individual words so that words that occur infrequently are worth more than words that occur frequently. This graph shows the IDF weighting value for a collection of 10,000 documents for total word counts ranging from 1 to 100.

lengths, one common extension is to normalize the document and query vectors, so that relevance becomes $\frac{\mathbf{d}^{\mathbf{T}}\mathbf{q}}{\|\mathbf{d}\|\|\mathbf{q}\|}$. This is the cosine of the angle between two points in an *n*-dimensional space.

Many extensions to similarity measures derive their power by finding a data-driven *weighting* for words. Each word of each document is re-scaled according to this weighting before the dot product or cosine measure is used to determine similarity. That is, if $\mathbf{f_i}$ is the count of the i^{th} word in a document, and $\mathbf{w_i}$ is the new weighting, $\mathbf{d_i} = \mathbf{f_i} \bullet \mathbf{w_i}$.

Perhaps the most commonly used weighting technique is Inverse Document Frequency (IDF) [Sparck-Jones 1972]:

$$\mathbf{w_i} = 1 + \log\frac{N}{\mathbf{n_i}}, \qquad \textbf{(2.1)}$$

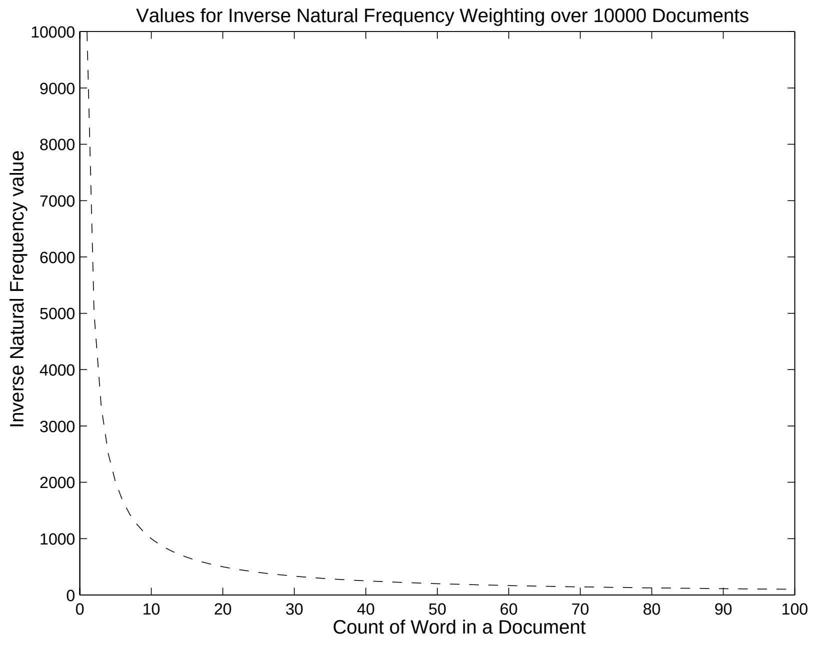

FIGURE 2-3. **Inverse Natural Word Frequency.**
Dividing a word count by its average frequency leads to numerical instability. This graph shows the weighting value for a collection of 10,000 documents for total word counts ranging from 1 to 100.

and its variants [Sparck-Jones 1979] [Croft and Harper 1979]:

$$\mathbf{w_i} = \log \frac{N - \mathbf{n_i}}{\mathbf{n_i}}, \tag{2.2}$$

where N is the total number of documents in the collection and $\mathbf{n_i}$ is the total number of occurrences of word i in the collection. This is similar to dividing each word by its average frequency:

$$\mathbf{w_i} = \frac{N}{\mathbf{n_i}}. \tag{2.3}$$

The average frequency is likely to be very close to zero for most words, leading to numerical instability. Among other things, IDF squashes the growth of the inverse natural frequency as it approaches zero.

Ultimately, these techniques reduce the weight of words that appear very frequently while greatly increasing the weight of words that appear very infrequently. Intuitively, this is desirable if there is more discriminating power in rare words than in highly used words, such as “the.” As an empirical matter, IDF and its variations rarely hurt performance and almost always improve it.

Another commonly used technique is based on so-called probabilistic models. This scheme usually involves pseudo-relevance feedback, which we will discuss in § 2.4. Another useful technique requires specially rescaling queries. Each word that is present in a query is first multiplied by its average frequency in those documents in which it appears [Kwok, 1996]. This brings the word frequencies of a query more in line with those of the collection. This works well for short unstructured queries, a topic we shall explore in detail in Chapter 6.

2.2.2 Adding Features

There are several possible additions to the word features that make up the basic VSM histogram. It is common to include components that count the occurrence of known compound words [Kwok, 1996; Sparck-Jones and Willett]. Similarly, commonly used noun phrases may also be added.

A related technique is *query expansion*. With this technique, a query is preprocessed so that the query includes not only all of its words, but their synonyms as well. The additional words are usually given a smaller weighting than they would otherwise. Query expansion is also used with relevance and pseudo-relevance feedback (see § 2.4).

2.3 Latent Semantic Indexing

VSM gains its computational advantages by sacrificing a great deal of document structure that may disambiguate meaning. Many techniques have been developed to improve the performance of VSM while retaining as much of its computational advantages as possible.

Latent semantic indexing (LSI) [Deerwester, *et al*, 1990] is one such technique. We mention it here because it has a machine learning flavor. LSI attempts to overcome the problems that hinder VSM by constructing a small matrix that retains only the most “important” information from the original document matrix. In particular, LSI uses the singular value decomposition (SVD) to construct this matrix. Briefly, the SVD of a matrix, **D**, is:

$$\mathbf{D} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathbf{T}},$$

where **U** contains orthonormal vectors, **V** contains orthonormal vectors and **S** is diagonal (see § A.5 for further discussion of properties and algorithms).

A natural interpretation of **U** is as the eigenvectors of the co-occurrence matrix. The co-occurrence matrix of **D** is:

$$(\mathbf{D}\mathbf{D}^{\mathbf{T}})_{\mathbf{ij}} = \sum_{\mathbf{k}} \mathbf{d}_{\mathbf{ki}}\mathbf{d}_{\mathbf{kj}}.$$

The co-occurrence matrix of **D** is a measure of the correlation between pairs of terms. It is similar to the covariance matrix (where mean values are first removed). The SVD of the co-occurrence matrix is:

$$\mathbf{D}\mathbf{D}^{\mathbf{T}} = \mathbf{U}\mathbf{S}^2\mathbf{U}^{\mathbf{T}}.$$

So, if **D** is the term-document matrix of a collection, **U** contains the eigenvectors of the co-occurrence matrix while each diagonal element of **S** is the square root of the corresponding eigenvalue. Each eigenvalue represents the contribution of its eigenvector to the variation of the data; higher values indicate more contribution. Thus, we can remove the *least* important factors by simply removing the eigenvectors with the smallest eigenvalues, creating a new set of matrices, $\hat{\mathbf{S}}$ and $\hat{\mathbf{U}}$. We can then use $\hat{\mathbf{S}}$ and $\hat{\mathbf{U}}$ as a new basis for **D**, projecting it into a lower dimensional space:

$$\hat{\mathbf{D}} = \hat{\mathbf{S}}^{-1}\hat{\mathbf{U}}^{\mathbf{T}}\mathbf{D}.$$

This operation results in a matrix of smaller size that provably represents the most variation in the original matrix. That is, if we project the smaller matrix back into the original space, the squared difference between the new matrix and the original will be minimized.

Queries are projected into the same low dimensional space and then compared using the cosine of the angle between the vectors in the new space. Other variations for LSI include

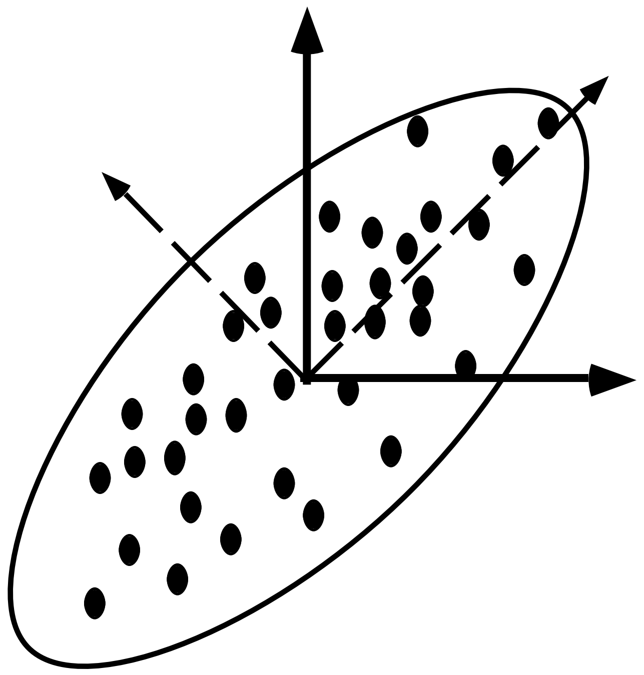

FIGURE 2-4. **An Example of Principal Components Analysis.**
Principal Components Analysis, a technique strongly related to latent semantic indexing, finds orthonormal projections. The first projection is the axis of maximum variation. The second projection is the axis of maximum variation orthogonal to the first, and so on. Here, the dashed axes represent the first two principal axes.

using the covariance matrix instead of the co-occurrence matrix using $\hat{\mathbf{S}}$ or $\hat{\mathbf{S}}^2$ instead of $\hat{\mathbf{S}}^{-1}$ to project documents and queries.

Though it has resisted a formal justification, experiments have shown that some scaled projection onto $\hat{\mathbf{U}}$ does sometimes improve retrieval performance. Hypotheses abound, including: 1) LSI removes noise from the document set, thus overcoming the problem of polysemy; 2) LSI finds synonyms or other meaningful underlying “topics” that are present in the collection, thus overcoming the problem of synonymy; and 3) LSI finds true clusters of documents (see [Hull, 1991] and [Deerwester, *et al*, 1990] for an extended discussion).

There is a strong similarity between LSI and principal components analysis (PCA), a dimensionality reduction technique that has been applied in a variety of settings (see FIGURE 2-4). In many tasks where PCA has been used, such as object recognition, it is used mainly to reduce computational complexity. In text retrieval, it is more often justified as a means to improve performance. We shall return to LSI in Chapter 4.

2.4 Relevance Feedback

Generally, information retrieval is an unsupervised process. We are given a set of documents and a query and we have to retrieve the best documents. We might imagine that performance could be improved if we had some indication of relevant and irrelevant items to use in ranking documents. It is sometimes possible to accomplish this by soliciting advice from users on-the-fly using *relevance feedback* [Salton and Buckley, 1990]. Relevance feedback adds extra iterations to the retrieval process. A query is presented to the system. The system returns the documents that it thinks matches the query. The user is then allowed to mark some of the documents as relevant and/or irrelevant. Those newly marked documents are then used to achieve better performance.

Some systems use the documents that are now *known* to be good as new queries into the database. Other documents that are relevant to these good documents are then returned. A more sophisticated approach is the Rocchio algorithm [Salton, 1971]. Here, a new query $\mathbf{q_r}$ is constructed:

$$\mathbf{q_r} = \frac{1}{|\mathbf{R}|}\sum_{\mathbf{D_i} \in \mathbf{R}} \mathbf{D_i} - \frac{1}{|\mathbf{S}|}\sum_{\mathbf{D_i} \in \mathbf{S}} \mathbf{D_i} \qquad \textbf{(2.4)}$$

where **R** is the set of known relevant documents and **S** the set of known non-relevant documents. The query $\mathbf{q_r}$ points towards the components the separate the relevant documents from the non-relevant documents. In practice, the negative components are removed from $\mathbf{q_r}$. Further, performance is improved by re-centering the new vector around the original, $\mathbf{q_o}$:

$$\mathbf{q_r} = \alpha\mathbf{q_r} + \beta\mathbf{q_o} \qquad \textbf{(2.5)}$$

[Singhal, 1997] introduces an extension called *query zoning*, where only the most egregiously misclassified elements of **S** are used. This tends to improve retrieval performance further.

So-called probabilistic models use relevance feedback to re-weight words in the original query (original work was done by [Maron and Kuhns 1960], but modern models are more strongly related to [Robertson and Sparck-Jones 1976]). There are various schemes for this re-weighting. Here is a standard one:

$$\mathbf{w_i} = \log \frac{\left(\frac{\mathbf{r_i}}{\mathbf{R} - \mathbf{r_i}}\right)}{\left(\frac{\mathbf{n_i} - \mathbf{r_i}}{\mathbf{N} - \mathbf{n_i} - \mathbf{R} + \mathbf{r_i}}\right)} \qquad \textbf{(2.6)}$$

where *N* is the number of documents in the collection, *R* is the number of relevant documents for the query, $\mathbf{n_i}$ the number of documents with word *i*, and $\mathbf{r_i}$ is the number of relevant documents with word *i*.

It is sometimes the case that user feedback is not feasible or desirable. It is possible to use *pseudo-relevance feedback*. In this case, it is assumed that the top few documents returned by the system are definitely relevant and the bottom few (or perhaps a few in the middle) are definitely irrelevant. The systems treats these as examples generated by a virtual user and proceeds from there. Pseudo-relevance feedback have enjoyed some success recently; however, it is not yet clear how well this technique works in general. In particular, it is important that the first few documents returned are relevant. If even a few bad documents are returned with high score, poor performance may result.

2.5 Tasks in Information Retrieval

The field of Information Retrieval is broad. There are several subareas within which researchers have focused their efforts. We have focused on the task where a system is given a set of documents, and whenever a user specifies a query, those documents are ranked by relevance. This is known as the *ad hoc* retrieval task. Here the goal is to find the best ranking method possible by whatever means. Documents are known beforehand and collections usually remain relatively unchanged.

In the *routing* and *filtering* tasks, a user has a set of standing information requests. New documents arrive regularly. The system receives a new document and decides whether it meets the criteria of those information requests and, if so, presents the document to the user.

For example, imagine that a user is subscribed to an AP news service, but is only interested in news items that are about natural disasters or events affecting Atlanta, GA. A good filtering system should pass along articles about earthquakes or the 1996 Olympics, but not about unrelated articles on the Chicago Bulls (unless, of course, the Atlanta Hawks defeat them). This task is strongly related to *text classification*, where one has a set of predefined classes and, given a new document, wants to determine class membership. Generally, classification (and recognition) algorithms are strongly supervised; that is, a system has access to many labelled examples beforehand and learns a model of class membership based on those labels. By contrast, the *ad hoc* task is often unsupervised.

There are other specific tasks of interest to the IR community. For example, how should one deal with data that are believed to be highly noisy and filled with many errors, such as text scanned in from faxes? There are other interesting special issues that arise from working with extremely large corpora, such as that on the size of the World Wide Web. In some domains, the collection is constantly changing, so algorithms that can efficiently update their data structures are of as much importance as algorithms that score documents accurately. There is increasing interest in cross-language retrieval, where queries may be given in one language, but are expected to retrieve documents in another language.

Finally, there are interesting retrieval issues in domains that do not include text at all, such as image retrieval, or sound classification. Although the tasks are similar, the structure of the data and the queries are often quite different. Each domain brings different challenges.

In this work, we are mostly concerned with issues that arise from text retrieval. Further, we are particularly interested in *ad hoc* retrieval tasks involving short queries.

2.6 Machine Learning and Information Retrieval

Machine learning has probably been applied most often in filtering and text classification. For example, [Lewis, 1992], [Lewis and Gale, 1994], [Cohen and Singer, 1996], [Lewis, et al, 1996], and [Schapire, Singer, and Singhal, 1998] all use machine learning to improve classification performance. Machine learning research has focused on classification tasks in general, so such techniques seem well-suited for these particular tasks.

Within the specific realm of *ad hoc* retrieval, machine learning has been applied less often. Although there has been some effort to apply machine learning to learn word weighting functions, the IR community has already expended a great deal of research effort over many decades in developing robust text-specific schemes, as we have seen in this chapter. The same can be said for reweighting schemes used in relevance feedback.

Latent semantic indexing can be seen as one attempt to use techniques that have found use in machine learning and apply them to text retrieval. What is of particular interest to us is that LSI is being used to “learn” not just a model of word importance for distinguishing between classes, but to learn a model that specifies general regularities about specific text collections.

2.7 Performance Measures

Determining how well a system performs is difficult. In this section we discuss several standard evaluation metrics and provide some examples of how they interact.

2.7.1 Precision and Recall

Many measures of retrieval performance have been proposed. The most commonly used are *precision* and *recall*. Precision is the ratio of relevant documents retrieved to the total number of documents retrieved. Recall is the ratio of relevant documents retrieved to the total number of relevant documents contained within the collection. Because systems provide an ordering on all documents for a given query, we can calculate precision and recall for the top *n* documents, with *n* ranging over the total number of documents in the collection. For example, in FIGURE 2-5 we have a collection made up of ten documents. For a particular query, five are relevant and five are not. If we examine only the first document returned, we can see that we have perfect precision (1.0) with recall equal to 0.2. Looking at the first four documents returned, we can see that three are relevant, resulting in a precision of 0.75 and a recall of 0.60. Precision and recall can be calculated for a single query as we have in our example, or averaged over many queries.

One usually wishes to measure performance in terms of both precision and recall. This is commonly done using a precision-recall graph. Precision is on the *y*-axis and recall on the

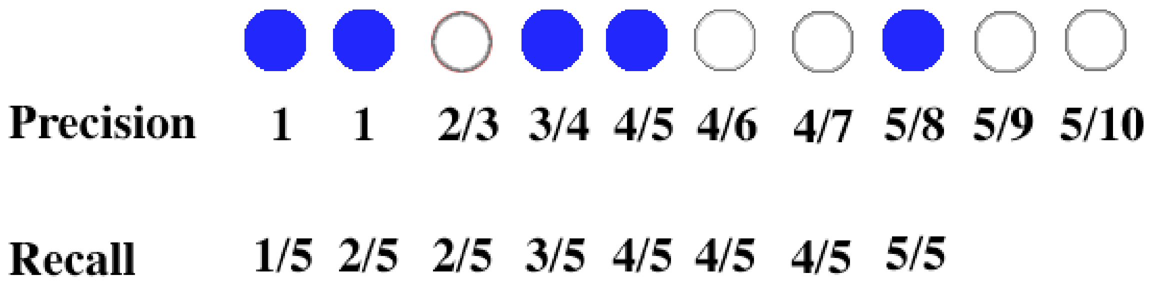

FIGURE 2-5. **Precision and Recall.**
Imagine a collection consisting of ten documents. For a particular query, five are relevant (solid) and five are irrelevant (outline). We can calculate the precision and recall when considering just the first document returned by the system, just the first two returned, first three, and so on.

x-axis. For our experiments, when several documents have the same score, precision and recall are calculated for all possible orderings of that subset and their average retained.

Generally speaking, precision and recall are inversely related. That is, as precision goes up, recall goes down and vice-versa. Thus, precision-recall curves have a slope of approximately -1, and one IR system is considered to have performed better than another when its precision-recall curve is above and to the right of the other. Naturally, a precision-recall graph only provides a qualitative measure of performance; however, the graph is often sufficient to determine at least whether one system is systematically outperforming another.

In any case, it is useful to have a single number to measure performance. Many have been suggested, including average precision, precision at a low number of documents and precision at a certain recall value. In average precision, precision is measured at every recall point and the average returned. This value is larger when relevant documents are ranked earlier and can lead to cases where a system with low recall outperforms a system with high recall because the former gets the first few documents right. Precision at a certain number of documents also favors systems that return good initial documents but does not focus on overall performance. Precision at a certain recall value is a measure of false positives; that is, how many documents one has to see before finding a certain number of relevant ones.

Other evaluation measures have been proposed to combine precision and recall into a single number. For example, [van Rijsbergen, 1979] suggests:

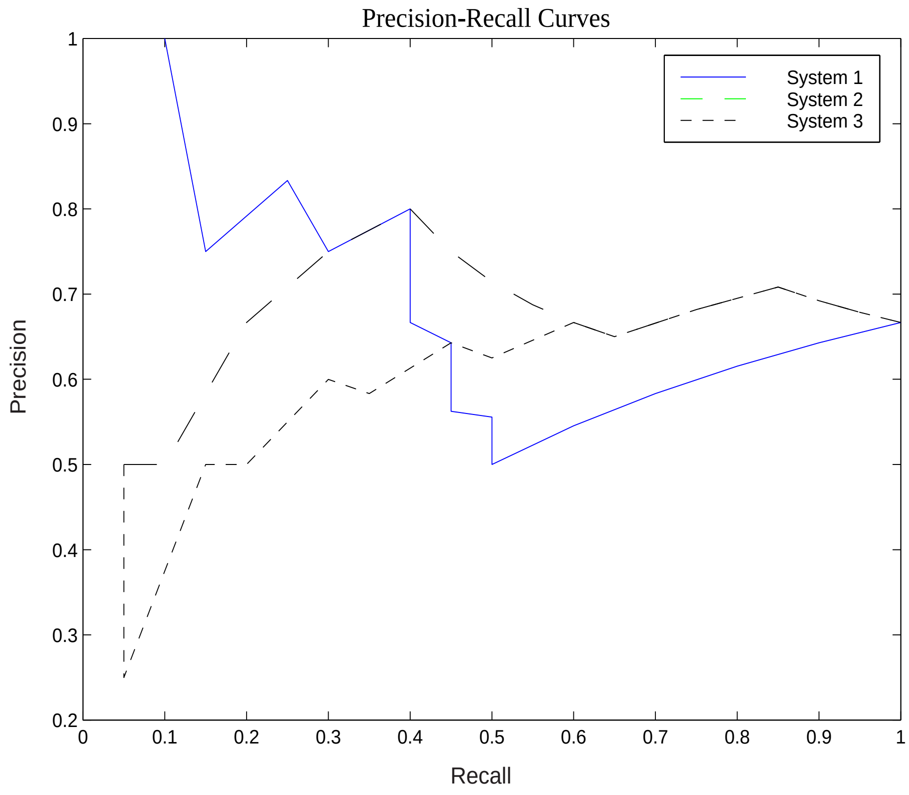

FIGURE 2-6. **Precision-Recall Curves for Three Systems**
Precision-Recall curves give a qualitative view of overall system performance. System 3 appears to have the worst overall performance while System 1 appears to have the best. See the text for more discussion.

$$E = 1 - \frac{(1 + b^2)PR}{b^2 P + R} \qquad \textbf{(2.7)}$$

where *P* is precision, *R* is recall, and *b* is the ratio of the importance of recall to precision. When *b*=10, recall is ten times more important than precision, but when *b*=0.1, recall is only a tenth as important as precision.

Let us build some intuition for some of these measures. Imagine three retrieval packages, `System-1`, `System-2` and `System-3`. On our set of queries, `System-1` always returns relevant documents first, but then begins returning many bad ones before finally returning the remaining relevant documents. `System-2`, like `System-1`, returns the relevant

	System 1	System 2	System 3
Precision at 13% of documents	0.75	0.50	0.25
Precision at 33% of documents	0.80	0.80	0.60
Precision at 66% of documents	0.50	0.65	0.65
Average Precision	0.67	0.67	0.60
Precision once 50% of relevant documents have been returned	0.56	0.71	0.62
Percentage of documents returned to achieve 50% recall	0.60	0.47	0.53

TABLE 2-1. **Performance According to Different Measures.**
Different measures reveal different behavior. System 3 is consistently worse on almost all measures than System 1 and 2; however, System 1 and 2 have very similar performance by several measures. See the text for more discussion.

documents early on about half the queries but performs poorly on the other half of the queries. `System-3` returns some relevant documents among the first third of the documents it returns, but in a random order. It then returns all the rest of the relevant documents so that only irrelevant documents are ranked the least relevant. FIGURE 2-6 and TABLE 2-1 show how these systems compare on some of the measures we have discussed.

The precision-recall curves reflect the specific biases of the systems. `System-1` does well at low points of recall, which in this case reflects that we are only considering the first few documents. On the other hand, because it returns several relevant documents only near the end, its performance dips quite a bit. While `System-2` has some similar behavior, the fact that it does not always return a relevant document first has devastating effects on the shape of its precision-recall curve. On some single-number measures, `System-1`'s advantages are not as clear. `System-1` and `System-2` have identical average precision scores, for example. In fact, `System-2` does better by some measures. Because it doesn't always give low scores to some relevant documents, it returns most of the relevant documents earlier than `System-1`.

These examples are somewhat atypical. The curves for `System-2` and `System-3` are rather unusual, resulting from the fact that these examples are hand-crafted. On the other hand, this does point out that various measures reveal different biases. In evaluating a system, it is important to know what features are most important. In this work, we will use many of these measures. We will focus on precision-recall curves; however, we will use one of the other criteria when we wish to highlight some particularly interesting systemic feature.

2.7.2 Real-Word Evaluation

Evaluating precision and recall can be difficult. Recall, in particular, requires manually determining for every query whether each document is relevant. The text retrieval community has developed several *instrumented collections*, containing not only documents, but sets of queries with appropriate relevance judgements. Unfortunately, accurate precision and recall values depend heavily upon the mechanism used to determine whether a document is actually relevant to a query. For extremely large collections, it is not even possible to pre-evaluate the relevance of all documents to all queries. In reality, a system might actually return the "true" documents, and still appear to perform poorly. In short, the process for determining relevance is necessarily noisy and flawed. By using the relevance metrics discussed in the previous section to measure a technique's performance, we are essentially trying to emulate that flawed process.

There are other issues as well. For example, queries might be biased towards certain kinds of results that do not exercise a system's strengths. Also, the notion of relevance is not really binary, as it is treated in the field. Clearly, some documents are more relevant than others and should be more highly valued.

In the end, a system has to be evaluated on whether it has done "the reasonable thing." Does Altavista really work? Certainly, one is able to find documents that one is interested in, but sometimes the number of false negatives is scandalously high. Does this even matter, given that there are probably several thousand relevant documents when only two are desired? "The reasonable thing" is difficult to quantify even if we are very specific about our goals. Nonetheless, it is the only real performance metric that matters. Recognizing this, we will sometimes include examples of what kinds of documents various techniques consider good without direct reference to their predetermined relevance.

2.8 Natural Language Understanding

Because we are dealing with natural language documents, it seems reasonable to apply techniques from natural language understanding and processing (NLU) to text retrieval. In fact, during the mid-1960s, IR and NLU were seen as part of the same field [Sparck-Jones and Willett, 1997]. The subfields grew apart during the 1970s in part because of the success of purely statistical techniques in IR.

Nevertheless, NLU techniques have not been ignored by IR researchers. For example, question parsing techniques have been used to better extract meaningful cues from queries in order to improve retrieval [Saracevic, *et al*, 1997], and natural language parsing has been used to build text summaries [Marsh, Hamburger and Grishman, 1997; Rau, 1997].

NLU techniques have also been used to build representations of documents that are very different than the vector space model. Some systems build graph-like models of text that capture both syntactic and semantic relationships [Rau and Jacobs, 1988; Mallery, 1991].

Unfortunately, techniques such as [Rau and Jacobs, 1988] are currently limited to constrained domains and relatively small texts. It is not clear that it is possible to incorporate into standard IR machinery those NLP techniques that either require domain-specific knowledge to be fully effective or carry a large computational burden.

Nevertheless, we might consider graph representations. Imagine that the nodes of a graph represent words and edges denote relationships between those words. Relationships might include common membership in a noun phrase, ownership, or may indicate subject-object interaction via a verb. For example, the sentence “Charles takes Parry’s basketball,”

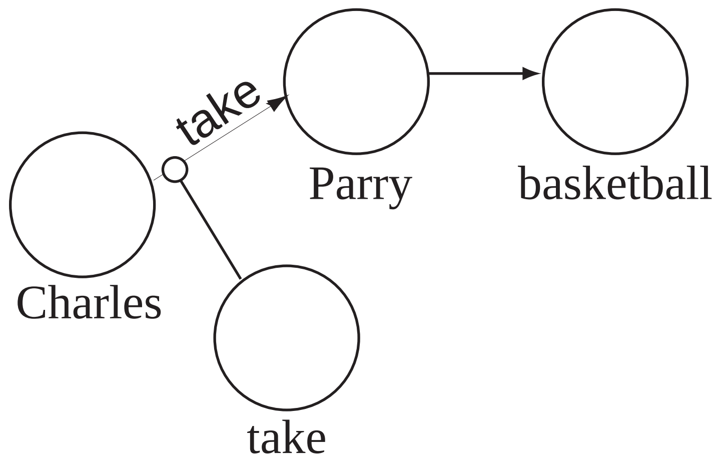

FIGURE 2-7. **A Graph of “Charles takes Parry’s basketball”.**
In this graph-based representation, a node indicates the presence of a word, while an arrow denotes an interaction of some kind, such as co-membership in a now phrase or a subject-object relationship via a transitive verb, such as “take.”.

might generate a graph like the one in FIGURE 2-7. With such a representation, we could treat retrieval as a form of graph matching.

We might expect that because the representation is richer, we would enjoy better performance. On the other hand, the Vector Space Model does capture a great deal of the graph’s structure. For example, nodes are much like the standard components of the vector representation. If “important” compound words are known or can be discovered, new “words” can be created to capture this relationship (for example, as described in § 2.2.2). By the same argument, objects that frequently act upon one another, such as “Charles” and “Parry’s basketball,” may also be combined and then treated simply as new words. In short, we can capture some of the graphs structure by treating graph nodes as vector components and edges as a kind of concatenation operator that creates new words.

Unfortunately, such a transformation may not reveal all the structure we are interested in or make certain operations as accurate or useful; however, the truth is that VSM is compact and easily manipulated by standard mathematical techniques. The latter is theoretically pleasing, but the former is equally as important, as we argue in the next section.

2.9 Computational Issues in Real-World Information Retrieval

The data sets used in text retrieval are large. Although real-world data sets may contain only 1000 documents consisting of about 10,000 different words, it is often the case that we are more interested in 100,000 or even 1,000,000 documents consisting of hundreds of thousands of distinct words. Even the smallest data sets are beyond the feasible reach of many machine learning algorithms.

There are several engineering challenges that must be addressed. First, simply storing and manipulating such data efficiently can be difficult. Fast algorithms are absolutely essential. Even algorithms polynomial in the size of the data are infeasible on serial machines. Further, bringing to bear statistical and machine learning techniques introduces more complexity. Such algorithms are usually at least polynomial in the size of the data to be learned, so even the smallest collections are beyond the reach of many machine learning algorithms. Clearly, a fully working retrieval system for something as large as the World Wide Web requires a systems-level engineering approach.

2.10 The Scope of This Work

There are several major issues and areas of interest in Information Retrieval. We have identified some of those in this chapter. In this work, we are concerned mainly with *ad hoc* retrieval. In particular, we are interested in systems—like many World Wide Web Search Engines—that will be faced with queries that mainly consist of a few words, as opposed to complete structured documents.

We are also explicitly not interested in addressing the filtering, routing and text classification tasks in this work. Further, we will not directly address how additional relevance feedback might affect our approach. Finally, while we will describe a parallel implementation that allows us to explore large collections in an interactive setting, we are not interested in building a robust user interface and addressing human-computer interaction issues. A user interface has been designed, but it is purely in the service of making experimentation easier and for demonstrating results. We acknowledge that these issues are both interesting and important; however, they remain beyond the primary goals of this work.