INFORMATION RETRIEVAL–1

CHAPTER OVERVIEW

The huge amount of information stored in electronic form, has placed heavy demands on information retrieval systems. This has made information retrieval an important research area. This chapter is concerned with the design of information retrieval systems. It discusses design features of systems and introduces various models, such as the classical (boolean, probabilistic, and vector space retrieval models), non-classical (information logic, situation theory, and interaction information retrieval model), and alternative (cluster, fuzzy, and LSI) models. A detailed discussion of vector space model is also given. The final topic of discussion in this chapter is information retrieval evaluation models.

31 INTRODUCTION

Information retrieval (IR) deals with the organization, storage, retrieval, and evaluation of information relevant to a user's query. A user in need of information formulates a request in the form of a query written in a natural language. The retrieval system responds by retrieving the document that seems relevant to the query. Research in IR is not new. It dates back to the 1960s when text retrieval systems were introduced. Traditionally, however, it is not considered an important application area of NLP. Interest in the field was generated due to the insurgence of the World Wide Web. The interaction between NLP and IR is now strengthening. Many NLP techniques, including the probabilistic model, have found application in IR systems and techniques such as latent semantic indexing, vector space retrieval, etc.

Traditionally, IR systems are not expected to return the actual information, only documents containing that information. As Lancaster (1979) pointed out:

Scanned by CamScanner

An information retrieval system does not inform (i.e., change the knowledge of) the user on the subject of her inquiry. It merely informs on the existence (or non-existence) and whereabouts of documents relating to her request.

The word *document* is a general term that includes non-textual information such as images and speech.

Our concern in this chapter is only with retrieval of text documents. This excludes question answering systems and data retrieval systems. The three systems differ in the nature of their queries and expected results of the queries. In both question answering systems and data retrieval systems, queries are very specific and precise in nature. On the other hand, queries submitted to IR systems are often vague and imprecise. A question answering system provides users with the answers to specific questions. Data retrieval systems retrieve precise data, usually organized in a well-defined structure. Unlike data retrieval systems, IR systems do not search for specific data. Nor do they search for direct answers to question, as question answering systems do.

2 DESIGN FEATURES OF INFORMATION RETRIEVAL SYSTEMS

Figure 9.1 illustrates the basic process of IR. It begins with the user's information need. Based on this need, he/she formulates a query. The IR system returns documents that seem relevant to the query. This is an engineering account of the IR system. The basic question involved is, 'what constitutes the information in the documents and the queries'. This, in turn is related to the problem of representation of documents and queries. The retrieval is performed by matching the query representation with document representation.

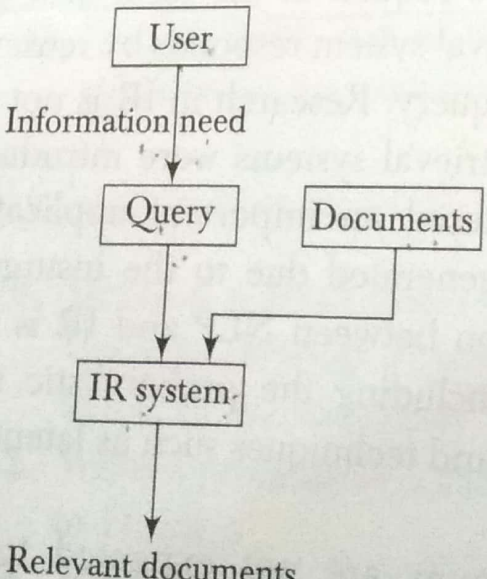

Figure 9.1 Basic information retrieval process

Scanned by CamScanner

The actual text of the document is not used in the retrieval process. Instead, documents in a collection are frequently represented through a set of index terms or keywords. In this chapter, the word *term* and *keyword* will be used independently. Keywords can be single word or multi-word phrases. They might be extracted automatically or manually (i.e., specified by a human). Such a representation provides a logical view of the document. The process of transforming document text to some representation of it is known as *indexing*. There are different types of index structures. One used data structure, commonly by the IR system, is the inverted index. An inverted index is simply a list of keywords, with each keyword carrying pointers to the documents containing that keyword. The computational cost involved in adopting a full text logical view (i.e., using a full set of words to represent a document) is high. Hence, some text operations are usually performed to reduce the set of representative keywords. The two most commonly used text operations are *stop word elimination* and *stemming*. Stop word elimination removes grammatical or functional words, while stemming reduces words to their common grammatical roots. *Zipf's law* can be applied to further reduce the size of index set. Not all the terms in a document are equally relevant. Some might be more important in conveying a document's content. Attempts have been made to quantify the significance of index terms to a document by assigning them numerical values, called weights. The choice of index terms and weights is a difficult theoretical and practical problem and several technique are used to cope with it. A number of *term-weighting* schemes have been proposed in the literature over the years. We discuss a few of them in this chapter.

9.2.1 Indexing

In a small collection of documents, an IR system can access a document to decide its relevence to a query. However, in a large collection of documents, this technique poses practical problems. Hence, a collection of raw documents is usually transformed into an easily accessible representation. This process is known as indexing. Most indexing techniques involve identifying good document descriptors, such as keywords or terms, which describe the information content of documents. A good descriptor is one that helps describe the content of the document and discriminate it from other documents in the collection. Luhn (1957, 1958) is considered the first person to advance the notion of automatic indexing of documents based on their content. He assumed that the frequency of certain word-occurrences in an article gave meaningful

Scanned by CamScanner

identification of the article's content. He proposed that the discrimination power of index terms is a function of the rank order of the frequency of their occurrence, and that middle frequency terms have the highest discrimination power. This model was proposed for the extraction of salient terms from a document. These extracted terms are then used to represent text. Thus, indexing is simply the representation of text (query and document) as a set of terms whose meaning is equivalent to some content of the original text.

As mentioned earlier, the word *term* can be a single word or multi-word phrases. For example, the sentence, *Design features of information retrieval systems*, can be represented as follows:

Design, features, information, retrieval, systems

It can also be represented by the set of terms:

Design, features, information retrieval, information retrieval systems

These multi-word terms can be obtained by looking at frequently appearing sequences of words, n-grams, part-of-speech tags, or by applying NLP to identify meaningful phrases or handcrafting. POS tagging helps extract meaningful sequences of words; it handles sense ambiguity, as words are assigned POS based on their local (sentential) context. Though statistical approaches to phrase extraction are more efficient, they fail to handle word order changes and structural variations, which are better handled by syntactic approaches (e.g., "junior school" vs "school junior"). In text retrieval conferences (TREC) conferences, the method used for phrase extraction is as follows:

1. Any pair of adjacent non-stop words is regarded a potential phrase.
2. The final list of phrases is composed of those pairs of words that occur in, say, 25 or more documents in the document collection.

The NLP is also used in the recognition of proper nouns and the normalization of noun phrases. Ideally, all names in the text need to be recognized and represented as a single entity, e.g. *President Kalam*, *President of India*, and variants of the same name are recognized as such. Phrase normalization captures structural variations in phrases. For example, the three phrases text categorization, categorization of text, and categorize text, are normalized to give text categorize.

9.2.2 Eliminating Stop Words

The lexical processing of index terms involves elimination of stop words. Stop words are high frequency words which have little semantic weight and are thus, unlikely to help in retrieval. These words play important

Scanned by CamScanner

grammatical roles in language, such as in the formation of phrases, but do not contribute to the semantic content of a document in a keyword-based representation. Such words are commonly used in documents, regardless of topics, and thus, have no topical specificity. Typical example of stop words are articles and prepositions. Eliminating them considerably reduces the number of index terms. The drawback of eliminating stop words is that it can sometimes result in the elimination of useful index terms, for instance the stop word *A* in *Vitamin A*. Some phrases, like *to be or not to be*, consist entirely of stop words. Eliminating stop words in such case, make it impossible to correctly search a document. Table 9.1 lists some of the stop words in English.

Table 9.1 Sample stop words in English

about	above	accordingly	across	after
afterwards	again	against	all	almost
alone	along	already	also	although
am	among	amongst	always	an
and	another	any	anybody	anyhow
anyone	anything	anywhere	apart	are
around	as	aside	at	away
awfully	be	because	been	before
beforehand	behind	being	below	beside
besides	best	better	between	beyond
both	brief	but	by	can
could	did	during	each	even
everybody	everyone	everything	everywhere	else
even	ever	every	for	from
further	had	has	have	her
herself	him	himself	he	furthermore
many	near	shall	she	self
whose	why	will	where	ex
except	far	first	five	former
formerly	over	overall	usually	appropriate

9.2.3 Stemming

Stemming normalizes morphological variants, though in a crude manner, by removing affixes from the words to reduce them to their stem, e.g., the words *compute*, *computing*, *computes*, and *computer*, are all be reduced to same word stem, *comput*. Thus, the keywords or terms used to represent text are stems, not the actual words. One of the most widely used stemming

Scanned by CamScanner

algorithms has been developed by Porter (1980). The stemmed representation of the text, *Design features of information retrieval systems*, is

{*design, featur, inform, retriev, system*}

Note that stop words have been removed in this representation and the remaining terms are in lower case.

One of the problems associated with stemming is that it may throw away useful distinctions. In some cases, it may be useful to help conflate similar terms, resulting in increased *recall*. In others, it may be harmful, resulting in reduced *precision* (e.g., when documents containing the term *computation* are returned in response to the query phrase *personal computer*). *Recall* and *precision* are the two most commonly used measures of the effectiveness of an information retrieval system, and are explained in detail later in this chapter.

9.2.4 Zipf's Law

Zipf made an important observation on the distribution of words in natural languages. This observation has been named Zipf's law. Simply stated, Zipf's law says that the frequency of words multiplied by their ranks in a large corpus is more or less constant. More formally,

$$\text{Frequent} \times \text{rank} \approx \text{constant}$$

This means that if we compute the frequencies of the words in a corpus, and arrange them in decreasing order of frequency, then the product of the frequency of a word and its rank is approximately equal to the product of the frequency and rank of another word. This indicates that the frequency of a word is inversely proportional to its rank. This relationship is shown in Figure 9.2.

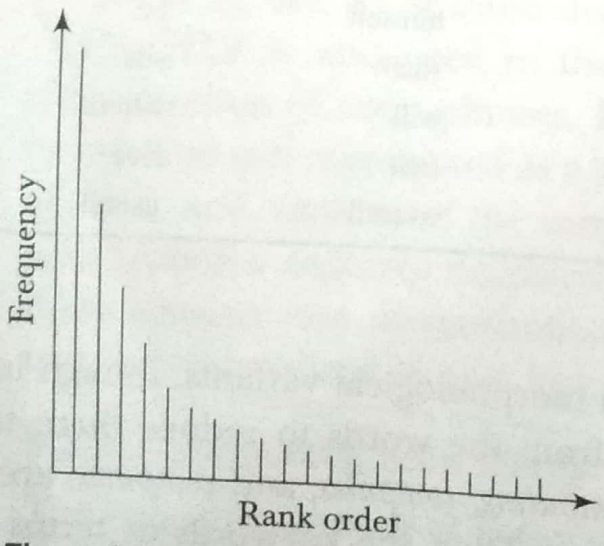

Figure 9.2 Relationship between the frequency of words and their rank order

Scanned by CamScanner

Empirical investigation of Zipf's law on large corpuses suggest that human languages contain a small number of words that occur with high frequency and a large number of words that occur with low frequency. In between, is a middling number of medium frequency terms. This distribution has important significance in IR. The high frequency words being common, have less discriminating power, and thus, are not useful for indexing. Low frequency words are less likely to be included in the query, and are also not useful for indexing. As there are a large number of rare (low frequency) words, dropping them considerably reduces the size of a list of index terms. The remaining medium frequency words are content-bearing terms and can be used for indexing. This can be implemented by defining thresholds for high and low frequency, and dropping words that have frequencies above or below these thresholds. Stop word elimination can be thought of as an implementation of Zipf's law, where high frequency terms are dropped from a set of index terms.

9.3 INFORMATION RETRIEVAL MODELS

An IR model is a pattern that defines several aspects of the retrieval procedure, for example, how documents and user's queries are represented, how a system retrieves relevant documents according to users' queries, and how retrieved documents are ranked. The IR system consists of a model for documents, a model for queries, and a matching function which compares queries to documents. The central objective of the model is to retrieve all documents relevant to a query. This defines the central task of an IR system.

Several different IR models have been developed. These models differ in the way documents and queries are represented and retrieval is performed. Some of them consider documents as sets of terms and perform retrieval based merely on the presence or absence of one or more query terms in the document. Others represent a document as a vector of term weights and perform retrieval based on the numeric score assigned to each document, representing similarity between the query and the document. These models can be classified as follows:

- Classical models of IR
- Non-classical models of IR
- Alternative models of IR

The three classical IR models—Boolean, vector, and probabilistic—are based on mathematical knowledge that is easily recognized and well understood. These models are simple, efficient, and easy to implement.

Scanned by CamScanner

Almost all existing commercial systems are based on the mathematical models of IR. That is why they are called classical models of IR.

Non-classical models perform retrieval based on principles other than those used by classical models, i.e., similarity, probability, and Boolean operation. These are best exemplified by models based on special logic technique, situation theory, or the concept of interaction.

The third category of IR models, namely alternative models, are actually enhancements of classical models, making use of specific techniques from other fields. The cluster model, fuzzy model, and latent semantic indexing (LSI) model are examples of alternative models of IR.

9.4 CLASSICAL INFORMATION RETRIEVAL MODELS

9.4.1 Boolean model

Introduced in the 50s, the Boolean model is the oldest of the three classical models. It is based on Boolean logic and classical set theory. In this model, documents are represented as a set of keywords, usually stored in an inverted file. An inverted file is a list of keywords and identifiers of the documents in which they occur. Users are required to express their queries as a Boolean expression consisting of keywords connected with Boolean logical operators (AND, OR, NOT). Retrieval is performed based on whether or not document contains the query terms.

Given a finite set

$$T = \{t_1, t_2, \ldots, t_i, \ldots, t_m\}$$

of index terms, a finite set

$$D = \{d_1, d_2, \ldots, d_j, \ldots, d_n\}$$

of documents and a Boolean expression—in a normal form—represent a query Q as follows:

$$Q = \wedge(\vee\theta_i),\ \theta_i \in \{t_i, \neg t_i\}$$

The retrieval is performed in two steps:

1. The set R_i of documents are obtained that contain or do not contain the term t_i:

   $$R_i = \{d_j \mid \theta_i \in d_j\},\ \theta_i \in \{t_i, \in t_i\}$$

   where $\neg t_i \in d_j$ means $t_i \notin d_j$
2. Set operations are used to retrieve documents in response to Q:

   $\cap R_i$

Example 9.1 Let the set of original documents be

$$D = \{D_1, D_2, D_3\}$$

Scanned by CamScanner

where

D_1 = Information retrieval is concerned with the organization, storage, retrieval, and evaluation of information relevant to user's query.

D_2 = A user having an information needs to formulate a request in the form of query written in natural language.

D_3 = The retrieval system responds by retrieving the document that seems relevant to the query.

Let the set of terms used to represent these documents be

T = {information, retrieval, query}

Then, the set D of document will be represented as follows:

$D = \{d_1, d_2, d_3\}$

where

d_1 = {information, retrieval, query}
d_2 = {information, query}
d_3 = {retrieval, query}

Let the query Q be

Q = information $\wedge$ retrieval

First, the sets S_1 and S_2 of documents are retrieved in response to Q, where

$S_1 = \{d_j \mid \text{information} \in d_j\} = \{d_1, d_2\}$
$S_2 = \{d_j \mid \text{retrieval} \in d_j\} = \{d_1, d_3\}$

Then, the following documents are retrieved in response to query Q

$\{d_j \mid d_j \in S_1 \cap S_2\} = \{d_1\}$

This results in the retrieval of the original document D_1 that has the representation d_1. If more than one document have the same representation, every such document is retrieved. Boolean information retrieval does not differentiate between these documents. With an inverted index, this simply means taking an intersection of the list of the documents associated with the keywords *information* and *retrieval.*

Boolean retrieval models have been used in IR systems for a long time. They are simple, efficient, and easy to implement and perform well in terms of recall and precision if the query is well formulated. However, the model suffers from certain drawbacks.

First, the model is not able to retrieve documents that are only partly relevant to user query; all information is 'to be or not to be'.

Second, a Boolean system is not able to rank the returned list of documents. It distinguishes between presence and absence of keywords but fails to assign relevance and importance to keywords in a document.

Scanned by CamScanner

Third, users seldom formulate their query in the pure Boolean expression that this model requires.

Numerous extensions of the Boolean model have been suggested to overcome these weaknesses. The best of these are the P-norm model developed by Salton et al. (1983) and a fuzzy-set model suggested by Paice (1984).

9.4.2 Probabilistic Model

The probabilistic model applies a probabilistic framework to IR. It ranks documents based on the probability of their relevance to a given query (Robertson and Jones 1976). Retrieval depends on whether probability of relevance (relative to a query) of a document is higher than that of non-relevance, i.e. whether it exceeds a threshold value. Given a set of documents D, a query q, and a cut-off value α, this model first calculates the probability of relevance and irrelevance of a document to the query. It then ranks documents having probabilities of relevance at least that of irrelevance in decreasing order of their relevance. Documents are retrieved if the probability of relevance in the ranked list exceeds the cut off value.

More formally, if $P(R/d)$ is the probability of relevance of a document d, for query q, and $P(I/d)$ is the probability of irrelevance, then the set of documents retrieved in response to the query q is as follows.

$$S = \{d_j \mid P(R/d_j) \geq P(I/d_j)\}\ P(R/d_j) \geq \alpha$$

Development of the probabilistic model was carried out largely by Maron and Kuhns (1960), Robertson and Sparck-Jones (1976), Robertson et al. (1982). Different mathematical methods for calculating the probabilities of relevance and irrelevance, as well as properties and applications, are discussed by Van Rijsbergen (1992), Callan et al. (1992), and Fur (1992).

Most of the systems assume that terms are independent when estimating probabilities for the probabilistic model. This assumption allows for accurate estimation of parameter values and helps reduce computational complexity of the model. However, this assumption seems to be inaccurate, as terms in a given domain usually tend to co-occur. For example, it is more likely that 'match point' will co-occur with 'tennis' rather than 'cricket'. Different forms of assumption are discussed in Robertson (1977). A comparison of the Boolean and probabilistic models can be found in Losee (1997). A comprehensive review of the probabilistic model is presented by Crestani et al. (1998). A baysian network version of the probabilistic model forms the basis of the INQUERY system (Callan et al. 1992).

Scanned by CamScanner

The probabilistic model, like the vector model, can produce results that partly match the user query. Nevertheless, this model has drawbacks, one of which is the determination of a threshold value for the initially retrieved set; the number of relevant documents by a query is usually too small for the probability to be estimated accurately.

9.4.3 Vector Space Model

The vector space model is one of the most well-studied retrieval models. Important contribution to its development was made by Luhn (1959), Salton (1968), Salton and McGill (1983), and van Rijsbergen (1977). The vector space model represents documents and queries as vectors of features representing terms that occur within them. Each document is characterized by a Boolean or numerical vector. These vectors are represented in a multi-dimensional space, in which each dimension corresponds to a distinct term in the corpus of documents. In its simplest form, each feature takes a value of either zero or one, indicating the absence or presence of that term in a document or query. More generally, features are assigned numerical values that are usually a function of the frequency of terms. Ranking algorithms compute the similarity between document and query vectors, to yield a retrieval score to each document. This score is used to produce a ranked list of retrieved documents. Given a finite set of n documents

$$D = \{d_1, d_2, \ldots, d_j, \ldots, d_n\}$$

and a finite set of m terms

$$T = \{t_1, t_2, \ldots, t_i, \ldots, t_m\}$$

each document is represented by a column vector of weights as follows:

$$(w_{1j}, w_{2j}, w_{3j}, \ldots, w_{ij}, \ldots, w_{mj})^t$$

where w_{ij} is the weight of the term t_i in document d_j. The document collection as a whole is represented by an $m \times n$ term-document matrix as

$$\begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1j} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2j} & \cdots & w_{2n} \\ w_{i1} & w_{i2} & \cdots & w_{ij} & \cdots & w_{in} \\ w_{m1} & w_{m2} & \cdots & w_{mj} & \cdots & w_{mn} \end{pmatrix}$$

Different term-weighting functions have been introduced (Salton and Buckley 1988). We discuss some of them in this section.

Scanned by CamScanner

Example 9.2 Consider the documents and terms in Example 9.1. Let the weights be assigned based on the frequency of the term within the document. Then, the associated vectors will be

(2, 2, 1)
(1, 0, 1)
(0, 1, 1)

The vectors can be represented as a point in Euclidean space, where the coordinates of point P_j are the components of d_j. The term-document matrix is

$$\begin{pmatrix} 2 & 1 & 0 \\ 2 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

This raw term frequency approach gives too much importance to the absolute values of various coordinates of each document. For example, a document with weights 4, 4, and 2, would be quite similar to document 1, except for the differences in magnitude of term weights.

To reduce the importance of the length of document vectors, we normalize document vectors. Normalization changes all vectors to a standard length. We convert document vectors to unit length by dividing each dimension by the overall length of the vector. Normalizing the term-document matrix shown in this example, we get the following matrix:

$$\begin{pmatrix} 0.67 & 0.71 & 0 \\ 0.67 & 0 & 0.71 \\ 0.33 & 0.71 & 0.71 \end{pmatrix}$$

Elements of each column are divided by the length of the column vector given by $\sqrt{\sum_i w_{ij}^2}$. The values shown in this matrix have been rounded to two decimal digits.

9.4.4 Term Weighting

Each term that is selected as an indexing feature for a document, acts as a discriminator between that document and all other documents in the corpus. Luhn (1958) attempted to quantify the discriminating power of the terms by associating the frequency of their occurrence (term frequency) within the document. He postulated that the most discriminating (content bearing) terms are mid frequency terms. This postulate can be refined by noting the following facts:

Scanned by CamScanner

1. The more a document contains a given word, the more that document is about a concept represented by that word.
2. The less a term occurs in particular document in a collection, the more discriminating that term is.

The first factor simply means that terms that occur more frequently represent the document's meaning more strongly than those occurring less frequently, and hence should be given high weights. In the simplest form, this weight is the raw frequency of the term in the document, as discussed earlier. The second factor actually considers term distribution across the document collection. Terms occurring in a few documents are useful for distinguishing those documents from the rest of the collection. Similarly, terms that occur more frequently across the entire collection are less helpful while discriminating among documents. This requires a measure that favours terms appearing in fewer documents. The fraction n/n_i—where n is the total number of the document in the collection and n_i the number of the document in which the term i occurs—exactly gives this measure. This measure assigns the lowest weight 1 to a term that appears in all documents and the highest weight of n to a term that occurs in only one document. As the number of documents in any collection is usually large, the log of this measure is usually taken, resulting in the following form of inverse document frequency (idf) term weight:

$$\text{idfi} = \log\left(\frac{n}{n_i}\right)$$

Inverse document frequency (idf) attaches more importance to more specific terms. If a term occurs in all documents in a collection, its idf is 0. Researchers have attempted to include term distribution in the weighting function (Sparck-Jones 1972, Salton 1971) to give a more accurate quantification of term importance. Sparck-Jones showed experimentally that a weight of $\log(n/n_i)+1$, termed as inverse document frequency, leads to more effective retrieval. Later researchers attempted to combine term frequency (tf) and idf weights, resulting in a family of tf × idf weight schemes having the following general form:

$$w_{ij} = \text{tf}_{ij} \times \log\left(\frac{n}{n_i}\right)$$

According to this scheme, the weight of a term t_i in document d_j is equal to the product of document frequency of the term and log of its inverse document frequency within the collection. The tf × idf weighting scheme combines both the 'local' and 'global' statistics to assign term weight. The tf component is a document specific statistic that measures

Scanned by CamScanner

the importance of a term within the document, whereas the idf is a global statistic that attempts to include distribution of the term across the document collection. These weighting schemes are well suited to the ad-hoc retrieval environment but pose problems in routing environment, where no fixed document collection exists. Usually, a training set of documents is used to compute statistics in such an environment and it is assumed that subsequent documents arriving at the system have the same statistical properties as the training set. We now give an example to explain how term weights can be calculated using the tf–idf weighting scheme.

Example 9.3 Consider a document represented by the three terms {tornado, swirl, wind} with the raw tf 4, 1, and 1 respectively. In a collection of 100 documents, 15 documents contain the term *tornado*, 20 contain *swirl*, and 40 contain *wind*. The idf of the term *tornado* can be computed as

$$\log\left(\frac{n}{n_i}\right) = \log\left(\frac{100}{15}\right) = 0.82$$

The idf of other terms are computed in the same way. Table 9.2 shows the weights assigned to the three terms using this approach.

Table 9.2 Computing tf-idf weight

Term	Frequency (tf)	Document frequency (n_i)	idf [$\log(n/n_i)$]	Weight (tf × idf)
Tornado	4	15	0.824	0.296
Swirl	1	20	0.699	0.699
Wind	1	40	0.398	0.389

Many variations of tf × idf measure have been reported. Some of these attempt to normalize tf and idf factors in different ways to allow for variations in document length (Salton and Buckley 1988). One way to normalize tf is to divide it by the frequency of the most frequent term in the document. This kind of normalization, often termed as maximum normalization, yields a value between 0 and 1. Normalization is needed because using absolute (raw) term frequency to weight terms favours longer documents over shorter ones. After frequency normalization, the weight of a term in a given document depends on the frequency of its occurrence in relation to the other terms in the same document, instead of its absolute frequency. Similarly, idf can be normalized by dividing it by the logrithm of the collection size (n).

$$w_{ij} = \frac{\mathrm{tf}_{ij}}{\max(\mathrm{tf}_{ij})} \times \log\left(\frac{n}{n_i}\right) / \log n$$

Scanned by CamScanner

A third factor that may affect weighting function is the document length. A term appearing the same number of times in a short document and in a long document, will be more valuable to the former. Most weighting schemes can thus be characterized by the following three factors:

- Within-document frequency or term frequency (tf)
- Collection frequency or inverse document frequency (idf)
- Document length

Any term weighting scheme can be represented by a triple ABC. The letter A in this triple represents the way the tf component is handled, *B* indicates the way the idf component is incorporated, and *C* represents the length normalization component. Possible options for each of the three dimensions of the triple are shown in Table 9.3. Different combinations of options can be used to represent document and query vectors. The retrieval model themselves can be represented by a pair of triples like nnn.nnn (doc = 'nnn', query = 'nnn'), where the first triple corresponds to the weighting strategy used for the documents and the second triple to the weighting strategy used for the query term.

Table 9.3 Calculating weight with different options for the three weighting factors

	Term frequency within document		
A	*n*	$tf = tf_{ij}$	Raw term frequency
	b	tf = 0 or 1 (binary weight)	
	a	$tf = 0.5 + 0.5\left(\frac{tf_{ij}}{\max tf \text{ in } D_j}\right)$	Augmented term frequency
	l	$tf = \ln(tf_{ij}) + 1.0$	Logarithmic term frequency
	L	$tf = \frac{\ln(tf_{ij} + 1.0)}{1.0 + \ln[\text{mean}(tf \text{ in } D_j)]}$	Average term frequency-based normalization
	Inverse document frequency		
B	*n*	wt = tf	No conversion
	t	$wt = tf \cdot \ln\left(\frac{n}{n_i}\right)$	Multiply tf with idf
	Document length		
C	*n*	$w_{ij} = wt$	(no conversion)
	c	w_{ij} is obtained by dividing each wt by sqrt [sum of (wts squared)]	

Scanned by CamScanner

There are many ways to compute each component. The simplest is to use either binary weight or raw term frequency. As shown in Table 9.3, the first occurrence of a term is more important than successive repeating occurrences. Thus, tf can be computed as $0.5 + 0.5$ (tf_{ij}/max tf in d_j) in which normalization is achieved by dividing tf by maximum tf value for any term in the document, or as $\ln(tf_{ij}) + 1.0$, which is known as logarithmic term frequency. The former computation is called augmented normalized term frequency. It causes tf to vary between 0.5 and 1. The problem with maximum normalization and augmented normalization of the tf component is that a single term in a document, with an unusually high frequency, may degrade the weights of the other terms significantly. However, this effect is not too pronounced with the augmented tf, because the highest frequency term cannot degrade the frequency of other terms below 0.5. The logarithmic term frequency reduces the effect of unusually frequent terms within a document, and also the importance of raw term frequency in a collection of documents with significant variations in length. It actually decreases the effect of all sorts of variations in tf, because for any two term frequencies tf_1 and $tf_2 > 0$, such that $tf_1 > tf_2$, the ratio of the logarithmic term frequencies will be always less than the ratio of the raw term frequencies, i.e.,

$$\frac{\log(tf_1)+1}{\log(tf_2)+1} < \frac{tf_1}{tf_2}$$

Different choices for *A*, *B*, and *C* for query and document vectors yield different retrieval modes, for example, ntc-ntc, lnc-ltc, etc. The choices for tf are *n* (use the raw term frequency), *b* (binary, i.e., neglect term frequency, term frequency will be 1 if term is present in the document, otherwise 0), *a* (augmented normalized frequency), *l* (logarithmic term frequency), and *L* (logarithmic frequency normalized by average term frequency). The options for idf are *n* (use 1.0, ignore idf factor) and *t* (use idf). The possible options listed in Table 9.3 for document length normalization are *n* (no normalization) and *c* (cosine normalization). To achieve cosine normalization, every element of the term weight vector is divided by the Euclidean length of the vector. This is called cosine normalization because the length of the normalized vector is 1 and its projection on any axis in document space gives the cosine of the angle between the vector and the axis under consideration.

The widely known weighting scheme, ntc-ntc, normalizes both the document and query term weight in the range 0–1 and may prove

Scanned by CamScanner

beneficial. The weighting scheme lnc-ltc means that document term weights are computed as the product of the logarithmic tf (*l*) of the given term, 1.0 (*n*) and cosine normalization (*c*) of the document vector. The query term weights are computed in the same way, except that each query term weight is also multiplied by the idf (t) of the given term in the document collection.

More recent weighting schemes integrate document length within the weighting formula yielding more complex retrieval models, for example, Okapi probabilistic search model (Robertson et al. 1995) and doc= "Lnu" model (Buckley et al. 1996). In TREC-3, the three systems with the best base performance were Okapi, INQUERY, and Cornell's Smart. The best performance was reported by Okapi system. Okapi uses the BM25 weighting algorithm introduced by developers of the probabilistic model during TREC-2 (Robertson et al. 1994) and TREC-3 (Robertson et al. 1995). Robertson and Walker (1994) developed the best match (BM) algorithms using the probabilistic model and some simple approximations to two-poisson model.

Considerable research efforts have been devoted to refining term weighting methods. As a result, a large number of term weighting schemes have been proposed in IR literature. (Salton and McGill 1983, Rijsbergen 1979). Some recent weighting schemes also consider the structure of the document.

A simple automatic method for obtaining indexed representation of the documents is as follows.

Step 1 Tokenization This extracts individual terms form a document, converts all the letters to lower case, and removes punctuation marks. The output of the first stage is a representation of the document as a stream of terms.

Step 2 Stop word elimination This removes words that appear more frequently in the document collection.

Step 3 Stemming This reduces the remaining terms to their linguistic root, to obtain the index terms.

Step 4 Term weighting This assigns weights to terms according to their importance in the document, in the collection, or some combination of both.

Table 9.4 shows the document vectors obtained after the application of these steps on sample documents shown in Figure 9.3.

Scanned by CamScanner

Document 1: Vector space model
Document 2: Probabilistic retrieval model
Document 3: Intelligent techniques in information retrieval

Figure 9.3 Sample documents

Table 9.4 Vector representation of sample documents after stemming

Stemmed terms	Document 1	Document 2	Document 3
inform	0	0	1
intellig	0	0	1
model	1	1	0
probabilist	0	1	0
retriev	0	1	1
space	1	0	0
technique	0	0	1
vector	1	0	0

9.4.5 Similarity Measures

Vector space model represents documents and queries as vectors in a multi-dimensional space. Retrieval is performed by measuring the 'closeness' of the query vector to document vector. Documents can then be ranked according to the numeric similarity between the query and the document. In the vector space model, the documents selected are those that are geometrically closest to the query according to some measure. The model relies on the intuitive notion that similar vectors define semantically related documents. Figure 9.4 gives an example of document and query representation in two-dimensional vector space. These dimensions correspond to the two index terms t_i and t_j. Document d_1 has two occurrences of t_i, document d_2 has one occurrence of t_i, and document d_3 has one occurrence of t_i and t_j each. Documents d_1, d_2, and d_3 are represented in this space using term weights—raw term frequency being used here—as coordinates. The angles between the documents and query are represented as θ_1, θ_2, and θ_3 respectively.

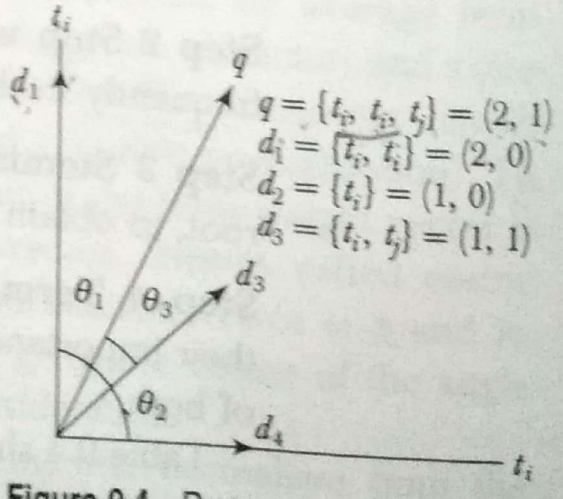

Figure 9.4 Representation in two-dimensional vector space

The simplest way of comparing document and query is by counting the number of terms they have in common. One frequently used similarity measure

Scanned by CamScanner

is to take the 'inner product' between the query and the document vector. The inner product is given by

$$\text{sim}\ (d_j, q_k) = (d_j, q_k) = \sum_{i=1}^{m} w_{ij} \times w_{ik}$$

where m is the number of terms used to represent documents in the collection.

Other measures, e.g., dice coefficient, Jaccard's coefficient, and cosine coefficient, attempt to normalize the similarity by the length of document and query. The dice coefficient defines the similarity between query k and document j as

$$\text{sim}\ (d_j, q_k) = \frac{2 \times \left(\sum_{i=1}^{m} w_{ij} \times w_{ik} \right)}{\sum_{i=1}^{m} w_{ij}^2 + \sum_{i=1}^{m} w_{ik}^2}$$

Jaccard's coefficient is defined as

$$\text{sim}\ (d_j, q_k) = \frac{\sum_{i=1}^{m} w_{ij} \times w_{ik}}{\sum_{i=1}^{m} w_{ij}^2 + \sum_{i=1}^{m} w_{ik}^2 - \sum_{i=1}^{m} w_{ij} \times w_{ik}}$$

The cosine measure is commonly used for measuring similarity in IR. It computes cosine of the angle between the document and query vector, to give a similarity value between 0 and 1. A minimum value of 0 (angle 90°) indicates that the vectors are unrelated (i.e., they have no terms in common), and a value of 1 means that the vectors share common terms. If d_j and q_k are the document and query vector respectively, then the cosine similarity is computed as

$$\text{sim}\ (d_j, q_k) = \frac{(d_j, q_k)}{\|d_j\| \|q_k\|} = \frac{\sum_{i=1}^{m} w_{ij} \times w_{ik}}{\sqrt{\sum_{i=1}^{m} w_{ik}^2} \times \sqrt{\sum_{i=1}^{m} w_{ij}^2}}$$

If both the document and the query vectors have been cosine-normalized, then the inner product yields the cosine similarity.

The cosine measure divides the numerator by the product of the length of vectors. This tends to give low similarities to long vectors, i.e. vectors with many terms. The overlap coefficient compensates for this by dividing

Scanned by CamScanner

by the vector having smaller sum of weights. More formally, this measure is defined as

$$\text{sim}\,(d_j, q_k) = \frac{\sum_{i=1}^{m} w_{ij} \times w_{ik}}{\min\left(\sum_{i=1}^{m} w_{ij}, \sum_{i=1}^{m} w_{ik}\right)}$$

9.5 NON-CLASSICAL MODELS OF IR

Non-classical IR models are based on principles other than similarity, probability, Boolean operations, etc., on which classical retrieval models are based. Examples include information logic model, situation theory model, and interaction model.

The *information logic model* is based on a special logic technique called logical imaging. Retrieval is performed by making inferences from document to query. This is unlike classical models, where a search process is used. Unlike usual implication, which is true in all cases except that when antecedent is true and consequent is false, this inference is uncertain. Hence, a measure of uncertainty is associated with this inference. The principle put forward by van Rijsbergen is used to measure this uncertainty. This principle says:

Given any two sentences x and y, a measure of the uncertainty of $y \rightarrow x$ *relative to a given data set is determined by the minimal extent to which one has to add information to the data set in order to establish the truth of* $y \rightarrow x$.

In fact, this model was developed in response to van Rijsbergen's realization that classical models were unable to enhance effectiveness and that new meaning-based models were required to do so.

The *situation theory* model is also based on van Rijsbergen's principle. Retrieval is considered as a flow of information from document to query. A structure called *infon*, denoted by ι, is used to describe the situation and to model information flow. An infon represents an n-ary relation and its polarity. The polarity of an infon can be either 1 or 0, indicating that the infon carries either positive or negative information.

For example, the information in the sentence, *Adil is serving a dish*, is conveyed by the infon

$$\iota = \langle\langle \textit{serving Adil, dish; 1} \rangle\rangle$$

The polarity of an infon depends on the support. The support of an infon is represented as $s \models \iota$ and means that the situation s makes the infon ι true. For example, the infon, $\iota = \langle\langle$ *serving Adil, dish*; 1$\rangle\rangle$ is made true by a situation $s1$ = "I see Adil serving a dish."

Scanned by CamScanner

A document d is relevant to a query q, if $d \models q$

If a document does not support the query q, it does not necessarily mean that the document is not relevant to the query. Additional information, such as synonymys, hypernyms/hyponyms, meronyms, etc., can be used to transform the document d into d' such that $d' \models q$. Semantic relationships in a thesaurus, like WordNet, are useful sources for this information. The transformation from d to d' is regarded as flow of information between situations.

The *interaction IR* model was first introduced in Dominich (1992, 1993) and Rijsbergen (1996). In this model, the documents are not isolated; instead, they are interconnected. The query interacts with the interconnected documents. Retrieval is conceived as a result of this interaction. This view of interaction is taken from the concept of interaction as realized in the Copenhagen interpretation of quantum mechanics. Artificial neural networks can be used to implement this model. Each document is modelled as a neuron, the document set as a whole forms a neural network. The query is also modelled as a neuron and integrated into the network. Because of this integration, new connections are built between the query and the documents, and existing connections are changed. This restructuring corresponds to the concept of interaction. A measure of this interaction is obtained and used for retrieval. Detailed mathematical treatments of the model have been discussed by Dominich (1992, 1993, 2001) and van Rijsbergen (1996).

9.6 ALTERNATIVE MODELS OF IR

9.6.1 Cluster Model

The cluster model is an attempt to reduce the number of matches during retrieval. The need for clustering was first pointed out by Salton. Before we discuss the cluster-based IR model, we would like to state the cluster hypothesis that explains why clustering could prove efficient in IR.

Closely associated documents tend to be relevant to the same clusters.

This hypothesis suggests that closely associated documents are likely to be retrieved together. This means that by forming groups (classes or clusters) of related documents, the search time reduced considerably. Instead of matching the query with every document in the collection, it is matched with representatives of the class, and only documents from a class whose representative is close to query, are considered for individual match.

Scanned by CamScanner

Clustering can be applied on terms instead of documents. Thus, terms can be grouped to form classes of co-occurrence terms. Co-occurrence terms can be used in dimensionality reduction or thesaurus construction. A number of methods are used to group documents. We discuss here, a cluster generation method based on similarity matrix. This method works as follows:

Let $D = \{d_1, d_2, \ldots, d_j, \ldots, d_m\}$ be a finite set of documents, and let $E = (e_{ij})_{n,n}$ be the similarity matrix. The element E_{ij} in this matrix, denotes a similarity between document d_i and d_j. Let T be the threshold value. Any pair of documents d_i and d_j $(i \neq j)$ whose similarity measure exceeds the threshold $(e_{ij} \geq T)$ is grouped to form a cluster. The remaining documents form a single cluster. The set of clusters thus obtained is

$$C = \{C_1, C_2, \ldots, C_k, \ldots, C_p\}$$

A representative vector of each class (cluster) is constructed by computing the centroid of the document vectors belonging to that class. Representation vector for a cluster C_k is

$$r_k = \{a_{1k}, a_{2k}, \ldots, a_{ik}, \ldots, a_{mk}\}$$

An element a_{ik} in this vector is computed as

$$a_{ik} = \frac{\sum_{d_j \in C_k} a_{ij}}{|C_k|}$$

where a_{ij} is weight of the term t_i, of the document d_j, in cluster C_k. During retrieval, the query is compared with the cluster vectors

$$(r_1, r_2, \ldots, r_k, \ldots, r_p)$$

This comparison is carried out by computing the similarity between the query vector q and the representative vector r_k as

$$s_{ik} = \sum_{i=1}^{m} a_{ik} q_i, \quad k = 1, 2, \ldots, p$$

A cluster C_k whose similarity s_k exceeds a threshold is returned and the search proceeds in that cluster.

Example 9.4

Let

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Scanned by CamScanner

be the term-by-document matrix. The similarity matrix corresponding to these documents is

1.0
0.9 1.0
0.4 0.4 1.0

Using a threshold of 0.7, we get the following two clusters:

$C_1 = \{d_1, d_2\}$
$C_2 = \{d_3\}$

The cluster vectors (representatives) for C_1 and C_2 are

$r_1 = (1\ 0.5\ 1\ 0\ 1)$
$r_2 = (0\ 0\ \ 1\ 1\ 0)$

Retrieval is performed by matching the query vector with r_1 and r_2.

9.6.2 Fuzzy Model

In the fuzzy model, the document is represented as a fuzzy set of terms, i.e., a set of pairs $[t_i, \mu(t_i)]$, where μ is the membership function. The membership function assigns to each term of the document a numeric membership degree. The membership degree expresses the significance of term to the information contained in the document. Usually, the significance values (weights) are assigned based on the number of occurrences of the term in the document and in the entire document collection, as discussed earlier. Each document in the collection

$$D = \{d_1, d_2, \ldots, d_j, \ldots, d_n\}$$

can thus be represented as a vector of term weights, as in the following vector space model

$$(w_{1j}, w_{2j}, w_{3j}, \ldots, w_{ij}, \ldots, w_{mj})^t$$

where w_{ij} is the degree to which term t_i belongs to document d_j.

Each term in the document is considered a representative of a subject area and w_{ij} is the membership function of document d_j to the subject area represented by term t_i. Each term t_i is itself represented by a fuzzy set f_i in the domain of documents given by

$$f_i = \{(d_j, w_{ij})\} \mid i = 1, \ldots, m;\ j = 1, \ldots, n$$

This weighted representation makes it possible to rank the retrieved documents in decreasing order of their relevance to the user's query.

Typically, queries are Boolean queries. For each term that appears in the query, a set of documents is retrieved. Fuzzy set operators are then applied to obtain the desired result.

Scanned by CamScanner

For a single-term query $q = t_q$, those documents from the fuzzy set $f_q = \{(d_j, w_{iq})\}$, are retrieved for which w_{iq} exceeds a given threshold. The threshold may also be zero.

Consider the case of an AND query $q = t_{q1} \wedge t_{q2}$

First, the fuzzy sets f_{q1} and f_{q2} are obtained and then, their intersection is obtained, using the fuzzy intersection operator $f_{q1} \vee f_{q2} = \min \{(d_j, w_{iq1}), (d_j, w_{iq})\}$

The documents in this set are returned.

Similarly, for an OR query $q = t_{q1} \wedge t_{q2}$, the union of fuzzy sets f_{q1} and f_{q2} is computed to retrieve documents as follows:

$$f_{q1} \vee f_{q2} = \max \{(d_j, w_{iq1}), (d_j, w_{iq})\}$$

Example 9.5 Consider the following three documents:

d_1 = {information, retrieval, query}
d_2 = {retrieval, query, model}
d_3 = {information, retrieval}

where the set of terms used to represent documents is

T = {information, model, query, retrieval}

The fuzzy sets induced by these terms are

$f_1 = \{(d_1, 1/3), (d_2, 0) (d_3, 1/2)\}$
$f_2 = \{(d_1, 0), (d_2, 1/3) (d_3, 0)\}$
$f_3 = \{(d_1, 1/3), (d_2, 1/3) (d_3, 0)\}$
$f_4 = \{(d_1, 1/3), (d_2, 1/3) (d_3, 1/2)\}$

If the query is $q = t_2 \wedge t_4$, then document d_2 will be returned.

9.6.3 Latent Semantic Indexing Model

Latent semantic indexing model is the application of single value decomposition to IR. The use of latent semantic indexing (LSI) is based on the assumption that there is some underlying 'hidden' semantic structure in the pattern of word-usage across documents, rather than just surface level word choice. LSI attempts to identify this hidden semantic structure through statistical techniques and use it to represent and retrieve information. This is done by modelling the association between terms and documents based on the manner in which terms co-occur across documents. LSI transforms the term-document vector space into a more compact latent semantic space. Each dimension in the reduced space corresponds to an 'artificial concept'. These concepts loosely correspond to a set of terms. It is believed that in the vector space of reduced dimensionality, the words referring to related concepts, i.e., words that

Scanned by CamScanner

co-occur, are collapsed into the same dimension. Latent semantic space is thus able to capture similarities that go beyond term similarity. In the latent semantic space, a query and a document can have high similarity even if the document does not contain a query term, provided the terms are semantically related.

Now we discuss how the LSI technique is actually employed in IR. The document collection is first processed to get a $m \times n$ term-by-document matrix, W, where m is the number of index terms and n is the total number of documents in the collection. Columns in this matrix represent document vectors, whereas the rows denote term vectors. The matrix element W_{ij} represents the weight of the term i in document j. The weight may be assigned based on term frequency or some combination of local and global weighting, as in the case of vector space model. Singular value decomposition (SVD) of the term-by-document matrix is then computed. Using SVD, the matrix is represented as a product of three matrices

$$W = TSD^T$$

where T corresponds to term vectors and has m rows and r columns and $r = \min(m, n)$. S corresponds to singular values. D^T is the transpose of D and has r rows and n columns. D corresponds to the document vector.

T and D are orthogonal matrices containing the left and right singular vectors of W. S is a diagonal matrix, containing singular values stored in decreasing order. We eliminate small singular values and approximate the original term-by-document matrix using truncated SVD. For example, by considering only the first k number of the largest singular values, along with their corresponding columns in T and D, we get the following approximation of the original term-by-document matrix in a space of k orthogonal dimensions, where k is sufficiently less than n:

$$W_k = T_k S_k D_k^T$$

where T_k is the first k columns of T, D_k^T is the first k columns of D^T, and S_k is the k largest singular values.

The matrix W_k is used for retrieval. The idea is that the elimination of small singular values throws out the 'noise' resulting from term usage variation, and captures the underlying 'hidden' semantic structure (i.e., concepts). Each dimension in the reduced space corresponds to artificial or derived concepts. Each such concept loosely represents a set of terms in the original term-document matrix. Documents with varying word usage patterns are collapsed to the same vector in k-space.

The queries are also represented in k-dimensional space. Let $q = (q_1, q_2, \ldots, q_m)$ be the original query vector, where each element q_i is the frequency

Scanned by CamScanner

of term i in the query q. The query q is represented in the k-dimensional space as

$$q_k = q^T T_k S_k^{-1}$$

where q^T is the transpose of the query vector, and T_k and S_k are the weights. $q^T T_k$ denotes the sum of k-dimensional term vectors and S_k^{-1}, the weights of each dimension. Thus, the query is represented as the weighted sum of its constituent term vectors.

Retrieval is performed by computing the similarity between query vector and document vector. For example, we can use the cosine similarity measure to rank documents to perform retrieval. In a keyword-based retrieval, relevant documents that do not share any term with the query are not retrieved. The LSI-based approach is capable of retrieving such documents, as similarity is computed based on the overall pattern of term usage across the document collection rather than on term overlap.

We now give an example to explain how a document in high-dimensional space is represented in a low, reduced, latent semantic space.

Example 9.6 Consider the matrix shown in Figure 9.5. This matrix defines five-dimensional space in which six documents, d_1, d_2, d_3, ..., d_6, have been represented. The five dimensions correspond to five index terms *tornado, storm, tree, forest,* and *farming*. For simplicity, tf has been used to weight index terms. Figure 9.6 shows the documents in a two-dimensional space. The vectors in the figure correspond to document vectors in the matrix R, which is the representation of X in reduced two-dimensional space. The two dimensions correspond to derived concepts obtained through the application of truncated SVD.

$$X = \begin{array}{l} \\ \text{tornado} \\ \text{storm} \\ \text{tree} \\ \text{forest} \\ \text{farming} \end{array} \begin{array}{c} \begin{array}{cccccc} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

Figure 9.5 A term-document matrix representing six documents in five-dimensional space

We now explain how to arrive at the reduced dimensionality representation of X. First, the SVD of X is computed to get the three matrices T, S, and D.

$$X_{5\times 6} = T_{5\times 5}\, S_{5\times 5}\, (D_{6\times 5})^T$$

These matrices are shown in Figures 9.7, 9.8, and 9.9 respectively. Consider the first two largest singular values of S, and rescale $D^T_{2\times 6}$ with singular

Scanned by CamScanner

values to get matrix $R_{2\times6} = S_{2\times2}D^T_{2\times6}$, as shown in Figure 9.10, where $S_{2\times2}$ is S restricted to two dimensions and $D^T_{2\times6}$ is D^T restricted to two columns. R is a reduced dimensionality representation of the original term-by-document matrix X and is used to plot the vectors in Figure 9.6.

To find out the changes introduced by the reduction, we compute document similarities in the new space and compare them with the similarities between documents in the original space. The document–document correlation matrix for the original n-dimensional space is given by the matrix $Y = X^TX$. Here, Y is a square, symmetric $n \times n$ matrix. An element Y_{ij} in this matrix gives the similarity between documents i and j. The correlation matrix for the original document vectors is shown in Figure 9.12. This matrix is computed using X, after normalizing the lengths of its columns. The document–document correlation matrix for the new space is computed analogously using the reduced representation R. Let N be the matrix R with length-normalized columns. Then, $M = N^TN$ gives the matrix of document correlations in the reduced space. The correlation matrix M is given in Figure 9.11. The similarity between document d_1, $d_4(-0.0304)$, and $d_6(-0.2322)$ is quite low in the new space because document d_1 is not topically similar to documents d_4 and d_6. In the original space, the similarity between documents d_2 and d_3 and between documents d_2 and d_5 is 0. In the new space, they have high similarity values (0.5557 and 0.8518 respectively) although documents d_3 and d_5 share no term with the document d_2. This topical similarity is recognized due to the co-occurrence of patterns in the documents.

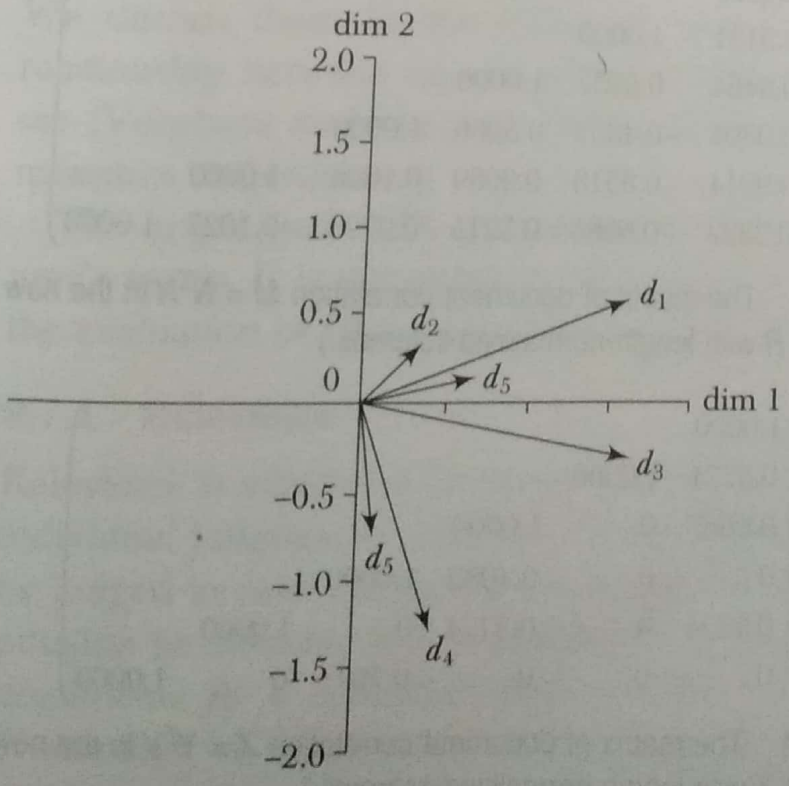

Figure 9.6 Documents in reduced two-dimensional space

Scanned by CamScanner

$$T = \begin{pmatrix} 0.3318 & 0.3338 & 0.8064 & -0.2426 & -0.2634 \\ 0.6693 & 0.1616 & -0.2737 & 0.5853 & -0.3293 \\ 0.5514 & 0.1038 & -0.0961 & -0.2667 & 0.7777 \\ 0.3583 & -0.5745 & -0.2148 & -0.5778 & -0.4021 \\ 0.0974 & -0.7223 & 0.4684 & 0.4400 & 0.2362 \end{pmatrix}$$

Figure 9.7 Matrix T for the SVD of the term-document matrix X shown in Figure 9.5

$$S = \begin{pmatrix} 2.3830 & 0 & 0 & 0 & 0 \\ 0 & 1.6719 & 0 & 0 & 0 \\ 0 & 0 & 1.2415 & 0 & 0 \\ 0 & 0 & 0 & 0.8288 & 0 \\ 0 & 0 & 0 & 0 & 0.5454 \end{pmatrix}$$

Figure 9.8 The matrix S for singular values of the SVD of the term-document matrix X

$$D^T = \begin{pmatrix} 0.6515 & 0.1392 & 0.6626 & 0.1912 & 0.2809 & 0.0409 \\ 0.3584 & 0.1996 & -0.1848 & -0.7756 & 0.0967 & -0.4320 \\ 0.3516 & 0.6495 & -0.4710 & 0.2042 & -0.2205 & 0.3773 \\ 0.0916 & -0.2927 & -0.3127 & -0.1662 & 0.7062 & 0.5309 \\ 0.3392 & -0.4829 & 0.0849 & -0.3042 & -0.6037 & 0.4330 \end{pmatrix}$$

Figure 9.9 The matrix D^T for singular values of the SVD of the term-document matrix

$$R = \begin{pmatrix} 1.5526 & 0.3318 & 1.5790 & 0.4557 & 0.6693 & 0.0974 \\ 0.5992 & 0.3338 & -0.3090 & -1.2967 & 0.1616 & -0.7223 \end{pmatrix}$$

Figure 9.10 The matrix $R_{2\times6} = S_{2\times2} D^T_{2\times6}$ representing documents in two-dimensional space

$$M = \begin{pmatrix} 1.0000 & & & & & \\ 0.9131 & 1.0000 & & & & \\ 0.8464 & 0.5557 & 1.0000 & & & \\ -0.0304 & -0.4353 & 0.5066 & 1.0000 & & \\ 0.9914 & 0.8518 & 0.9089 & 0.1008 & 1.0000 & \\ -0.2322 & -0.6086 & 0.3215 & 0.9793 & -0.1027 & 1.0000 \end{pmatrix}$$

Figure 9.11 The matrix of document correlation $M = N^T N$ in the new space (N is matrix R with length-normalized columns.)

$$Z = \begin{pmatrix} 1.0000 & & & & & \\ 0.5774 & 1.0000 & & & & \\ 0.6667 & 0 & 1.0000 & & & \\ 0 & 0 & 0.4082 & 1.0000 & & \\ 0.5774 & 0 & 0.5774 & 0 & 1.0000 & \\ 0 & 0 & 0 & 0.7071 & 0 & 1.0000 \end{pmatrix}$$

Figure 9.12 The matrix of document correlation $Z = Y^T Y$ in the new space (Y is matrix X with length-normalized columns.)

Scanned by CamScanner

The LSI performs IR based on concept. It is completely automatic and has been applied successfully (Deerwester et al. 1990, Foltz 1990) in many IR systems. However, it is costly in terms of computation.

9.7 EVALUATION OF THE IR SYSTEM

The evaluation of IR systems is the process of assessing how well a system meets the information needs of its users (Voorhees 2001). Evaluating an IR system is a difficult task involving a number of areas including cognition, statistics, and man-machine interactions. IR evaluation models can be broadly classified as system driven models and user-centered models. System driven models (Cleverdon et al. 1966) measure how well a system ranks documents; user-centered models measure user satisfaction. Cleverdon listed the following six criteria that can be used for evaluation:

1. *Coverage of the collection*: The extent to which the system
2. *Time lag*: The time that elapses between submission of a query and getting back the response
3. *Presentation format*
4. *User effort*: The effort made by the user to obtain relevant information
5. *Precision*: The proportion of retrieved documents that are relevant
6. *Recall*: The proportion of relevant documents that are retrieved

Of these criteria, recall and precision have most frequently been applied in measuring IR. Both are related to effectiveness, i.e., the ability of a system to retrieve relevant documents in response to user query. A number of effectiveness measures have been formulated (van Rijsbergen 1979). We discuss them in the following section. To better understand the relationship between aspects of retrieval process and different measures, see (Voorhees and Harman 1999), where correlations between pairs of measures are estimated.

The major goal of IR is to search for documents that are relevant to a user's query. It is necessary to understand what constitutes relevance, as the evaluation of IR systems relies on the notion of relevance.

9.7.1 Relevance

Relevance is subjective in nature (Saracevic 1991), i.e., it depends on the individual judgements of users. Given a query, the same document may be judged as relevant by one user and non-relevant by another. It is not possible to measure this 'true relevance' because no human can read all documents in a collection and provide a relevance assessment. Most evaluations of IR systems have so far been done on test document

Scanned by CamScanner

collections with known relevance judgments. These test document collections contain documents from a particular discipline; for a set of questions representing information needs, relevance assessments are obtained from experts of that discipline. This provides an experimental setup for evaluating the performance of a retrieval strategy. If a retrieval strategy performs well under these situations, it is expected to perform well in an operational environment where relevance is not known.

Another issue with relevance is the degree of relevance. Traditionally, relevance has been visualized as a binary concept, i.e., a document is either relevant or not relevant; whereas relevance is actually a continuous function (a document may be exactly what the user wants or it may be closely related). This is an attractive but difficult proposition and current evaluation techniques do not support it.

A number of relevance frameworks have been proposed by Saracevic (1996). This includes system, communication, psychological, and situational frameworks. The most inclusive of these is the situational framework, which is based on a cognitive view of the information seeking process and considers the importance of situation, context, multi-dimensionality, and time. A survey of relevance studies has been discussed by Mizzarro, (1996).

9.7.2 Effectiveness Measures

Effectiveness is purely a measure of the ability of a system to satisfy the user in terms of the relevance of documents retrieved (Rijsbergen 1979). Aspects of effectiveness include whether the documents returned are relevant to the user, whether they are presented in order of relevance, whether a significant number of relevant documents in the collection are returned to the user, etc. A number of measures have been proposed to quantify effectiveness. As stated earlier, the most commonly used measures of effectiveness are precision and recall. These measures are based on relevance judgments.

Precision and Recall

Precision is defined as the proportion of relevant documents in a retrieved set. This can be seen as the probability that a relevant document is retrieved. *Recall* is the proportion of relevant documents in a collection that have actually been retrieved. Precision measures the accuracy of a system while recall measures its exhaustiveness. Precision and recall can be computed as follows:

Scanned by CamScanner

$$\text{Precision} = \frac{\text{Number of relevant document retrieved } (NR_{ret})}{\text{Total number of documents retrieved } (N_{ret})}$$

$$\text{Recall} = \frac{\text{Number of relevant documents retrieved } (NR_r)}{\text{Total number of relevant documents in the collection } (NR_{rel})}$$

These definitions of precision and recall are based on binary relevance judgment, which means that every retrievable item is recognizably 'relevant', or recognizably 'not relevant'. Hence, for every search result, all retrievable documents will be either (i) relevant or non-relevant and (ii) retrieved or not retrieved. Thus, each document will fall into one, and only one, of four cells of the matrix, as sown in Figure 9.13. This matrix is used to derive a number of measures.

	Relevant	Non-relevant	
Retrieved	$A \cap B$	$\bar{A} \cap B$	B
Not-retrieved	$A \cap \bar{B}$	$\bar{A} \cap \bar{B}$	$\bar{B}$
	A	$\bar{A}$	

Figure 9.13 Relevant matrix

Referring to Figure 9.13, precision and recall will be given as follows:

$$\text{Precision} = \frac{|A \cap B|}{|B|} = \frac{NR_{ret}}{N_{ret}}$$

$$\text{Recall} = \frac{|A \cap B|}{|A|} = \frac{NR_{ret}}{NR_{rel}}$$

where A = Set of relevant documents
$|A|$ = No. of relevant documents in the collection (NR_{rel})
B = Set of retrieved documents
$|B|$ = No. of retrieved documents (NR_{ret})

It is clear from the preceding definitions that the total number of relevant documents in a collection must be known in order for recall to be calculated. The amount of effort and time required from the user makes this almost impossible in most operating environment. To provide a framework of evaluation for IR systems, a number of test collections have been developed (Cranfield and TREC). These collections are accompanied by a set of queries and relevance judgements. These test

Scanned by CamScanner

collections make it possible for IR researchers to efficiently evaluate their experimental approaches and compare the effectiveness of their system with that of others. In Table 9.5, basic statistics for a number of test collections are presented.

Table 9.5 IR test collections

Collection	Number of documents	Number of queries
Cranfield	1400	225
CACM	3204	64
CISI	1460	112
LISA	6004	35
TIME	423	83
ADI	82	35
MEDLINE	1033	30
TREC-1	742,611	100

There exists a trade-off between precision and recall, though a high value of both at the same time is desirable. The trade-off is shown in Figure 9.14. Precision is high at low recall values. As recall increases, precision decreases. The ideal case of perfect retrieval requires that all relevant documents be retrieved before the first non-relevant document is retrieved. This is shown in the figure by the line parallel to *x*-axis having a precision of 1.0 at all recall points. Recall is an additive process. Once the highest recall (1.0) is achieved, it remains 1.0 for any subsequent document retrieved. We can always achieve 100% recall by retrieving all documents in the collection, but this defeats the intent of an IR system.

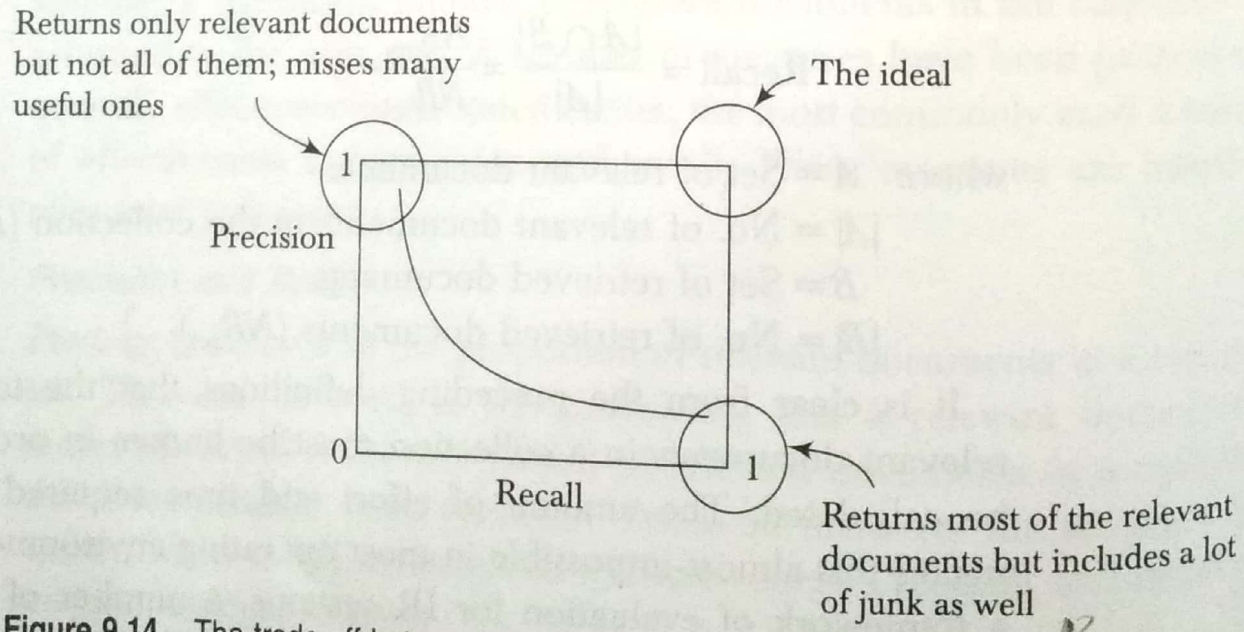

Figure 9.14 The trade-off between recall and precision

Scanned by CamScanner

A number of researchers have discussed the relationship between recall and precision (Cleverdon 1972, Robertson 1975, Gordon and Kochen 1989, Buckland and Gey 1994). Some of them modelled precision and recall as continuous functions (Robertson 1975, Gordon and Kochen 1989), while others (Bookstein 1974) described recall and precision in terms of two-Poisson discrete model. Buckland and Gey (1994) studied the relationship between precision and recall, and suggested that a two-stage, or more generally, a multi-stage retrieval procedure is likely to achieve the goal of improving both precision and recall simultaneously, even though the trade-off between them cannot be avoided.

In order to evaluate the performance of an IR system, recall and precision are almost always used together. One measure is to calculate precision at a particular cut-off. Typical cut-offs are 5 documents, 10 documents, 15 documents, etc.

Yet another measure is *non-interpolated average precision*, which is average of the precision at observed recall points. Observed recall points correspond to points where a relevant document is retrieved. We first compute precision at each point where a relevant document is found and then compute average of these precision numbers to get a single number. Precision at relevant documents not in the returned set is assumed to be zero. We now give an example to illustrate how precision is calculated.

Table 9.6 An example of retrieval

Rank	Document #	Relevance
1	10	x
2	8	x
3	5	
4	3	
5	1	x
6	2	
7	4	
8	7	
9	9	x
10	6	x

Example 9.7 Table 9.6 shows the ranking of 10 documents for a particular retrieval. The crossed documents are those that are relevant. Let the total number of relevant document be five.

Precision values at 5 and 10 documents are given as follows:

Precision at 5 $3/10 = 0.3$

Precision at 10 $5/10 = 0.5$

Scanned by CamScanner

Non-interpolated Average Precision The observed recall points are 0.2, 0.4, 0.6, 0.8, and 1.0. These recall values correspond to the documents marked relevant in the table. We have one of five relevant documents retrieved after retrieving only one document. This correspond to a recall value of $1/5 = 0.2$. After two documents, the recall is $2/5 = 0.4$. As the third document retrieved is not relevant, recall value does not change. The next relevant document is found after five documents have been retrieved, resulting in a recall value of $3/5 = 0.6$. Similarly, the other two recall points are calculated. The precision values at these points are as follows:

Precision at recall point 0.2: $1/1 = 1.0$
Precision at recall point 0.4: $2/2 = 1.0$
Precision at recall point 0.6: $3/5 = 0.6$
Precision at recall point 0.8: $4/9 = 0.4$
Precision at recall point 1.0: $5/10 = 0.5$
Non-interpolated average precision = 0.7

Considering the fact that a system may not always retrieve all the relevant documents, and that the number of relevant documents is not the same for all queries, precision values are interpolated for a set of recall points. The most widely used recall levels are 0.0, 0.1, 0.2, 0.3... 1.0. The precision values are calculated at each of these 11 recall levels and then averaged to get a single value. This is known as 11-point interpolated average precision. This has become almost a standard in evaluating the performance of an IR system. The interpolation used at TREC states that, precision at a given recall level is the greatest known precision at any recall level greater than or equal to this given level. For example, if the observed recall points are 0.25, 0.4, 0.55, 0.8, and 1.0, then precision at recall level 0.3 will be the maximum of the precision at recall levels 0.4, 0.55, 0.8, and 1.0, and not precision at recall point 0.4 where the 30% recall (i.e., recall level 0.3) is first reached. The interpolated precision at standard recall points for the documents shown in the Table 9.5 is computed in Example 9.8.

Example 9.8 Consider the following precision values at observed recall points:

0.25	1.0
0.4	0.67
0.55	0.8
0.8	0.6
1.0	0.5

Scanned by CamScanner

The interpolated precision at the standard 11 recall levels will be

0.0	1.0
0.1	1.0
0.2	1.0
0.3	0.8
0.4	0.8
0.5	0.8
0.6	0.6
0.7	0.6
0.8	0.6
0.9	0.5
1.0	0.5

Interpolated average precision = 0.745

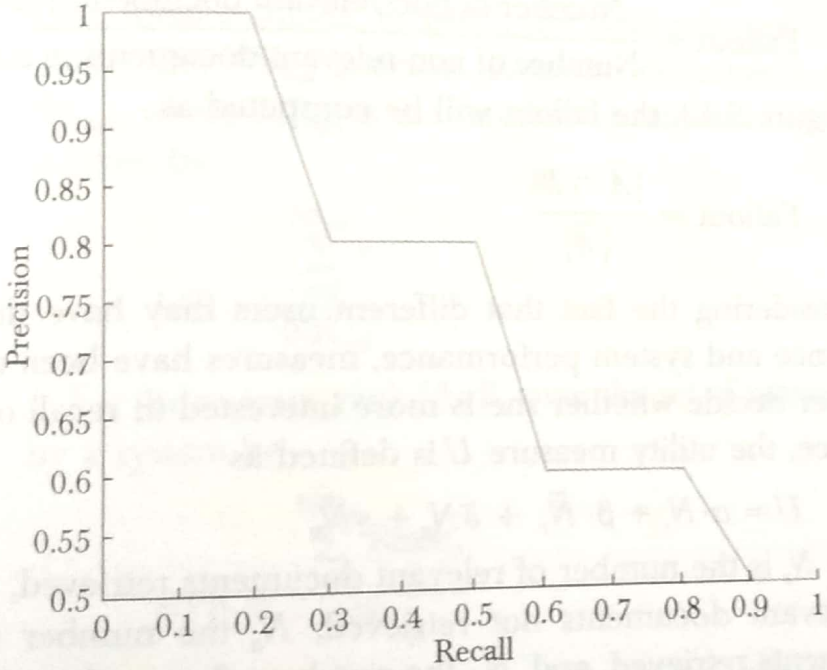

Figure 9.15 Recall–precision curve for interpolated precision

Often, precision values are calculated at different recall levels and a recall–precision graph, like the one shown in Figure 9.15, is plotted. As a retrieval system is evaluated over several queries, such a graph is usually plotted using precision figures averaged over all queries (Salton and McGill 1983, van Rijsbergen 1979). The most standard method for deriving a recall–precision graph is to plot the average over all queries in the interpolated precision values for a set of 11 standard recall points, namely 0.0, 0.1, 0.2, 0.3, ..., 1.0.

Instead of the recall–precision graph, the mean average precision is sometimes used to evaluate an IR system. The non-interpolated average

Scanned by CamScanner

precision is averaged over all queries to get the mean average precision (MAP). Geometrically, MAP is the area below the non-interpolated recall–precision curve (Voorhees and Harman 1999). The 11-point interpolated average precisions (11 avgP) can also be used for calculating MAP, though the non-interpolated measure has the advantage that it rewards systems that quickly retrieve (give high ranks to) relevant documents.

The R-precision is the precision after a total number of R documents relevant to the query have been retrieved.

Recall is not defined if there is no relevant document in a collection. An alternative measure is fallout, which may be seen as the inverse of recall. It is not defined only if all the documents in the collection are relevant (Salton 1983). Fallout is the ratio of non-relevant documents retrieved to non-relevant documents in the collection.

$$\text{Fallout} = \frac{\text{Number of non-relevant documents retrieved } (N_n)}{\text{Number of non-relevant documents in the collection}}$$

For Figure 9.13, the fallout will be computed as

$$\text{Fallout} = \frac{|\bar{A} \cap B|}{|\bar{A}|}$$

Considering the fact that different users may have different ideas of relevance and system performance, measures have been developed to let the user decide whether she is more interested in recall or precision. For instance, the utility measure U is defined as

$$U = \alpha \cdot N_r + \beta \cdot \bar{N}_r + \delta N_n + \gamma \bar{N}_n$$

where N_r is the number of relevant documents retrieved, $\bar{N}_r$ the number of relevant documents not retrieved, N_n the number of non-relevant documents retrieved, and $\bar{N}_n$ the number of non-relevant documents not retrieved (α, β, δ, and γ are positive weights specified by the user). This measure was later simplified by considering only retrieved documents.

The F-measure takes into account both precision and recall. It is defined as the harmonic mean of recall and precision.

$$F = \frac{2PR}{P + R}$$

Compared to the arithmetic mean, both recall and precision need to be high for harmonic mean to be high.

The E-measure is a variant of the F-measure. It allows weighting to emphasize precision rather than recall. It is defined as

Scanned by CamScanner

$$E = \frac{(1+\beta^2)PR}{\beta^2 P + R} = \frac{(1+\beta^2)}{\frac{\beta^2}{R} + \frac{1}{P}}$$

where P is precision, R is recall, and β is the relative importance of P compared to R. The value of β controls the trade-off between precision and recall. Setting β to 1 gives equal weight to precision and recall, resulting in a harmonic mean of recall and precision ($E = F$). $\beta > 1$ gives more weight to precision, and $\beta < 1$ gives more weight to recall.

Swets (1969) developed a model that received attention in the literature (Heine 1974; Bookstein 1977). None of these alternative measures, however, received as widespread acceptance as did the recall–precision model.

Normalized recall measures how close the set of retrieved documents is to an ideal retrieval, in which the most relevant NR_{rel} document appears in the first NR_{rel} position. Relevant documents are ranked 1, 2, 3, ..., NR_{rel}, where NR_{rel} is the number of relevant documents. The ideal rank is given by

$$IdR = \frac{\sum_{r=1}^{NR_{rel}} r}{NR_{rel}}$$

Let the average rank (AvR) over the set of relevant documents retrieved by a system be

$$AvR = \frac{\sum_{r=1}^{NR_{rel}} Rank_r}{NR_{rel}}$$

where $Rank_r$ represents the rank of the rth relevant document. The difference between AvR and IdR given by AvR – IdR, represents a measure of the effectiveness of the system. This difference can range from 0, for the perfect retrieval (AvR – IdR), to ($N - NR_{rel}$), for the worst case (N is the total number of documents in the collection). The worst case is when all the N documents are retrieved and the relevant documents, NR_{rel}, are the last retrieved. The expression AR–IR can be normalized by dividing it by ($N - NR_{rel}$) and then subtracting the result from 1. The normalized recall (NR) is given by

$$NR = 1 - \frac{AvR - IdR}{N - NR_{rel}}$$

Scanned by CamScanner

This measure ranges from 1 for the best case, to 0 for the worst case. If the value of NR is close to 1, the ranks of relevant documents in average case deviate very little from the ideal case. A high value of NR indicates that the ranks of the relevant documents in the average case deviate considerably from the ideal case.

9.7.3 User-centred Evaluation

The system-driven model is still the dominant approach followed in IR research for evaluation of IR systems. The evaluation here, is made on a test collection having known relevance judgments. These relevance judgements were usually provided by problem domain experts, and are binary, objective, topical, and static in nature, and lack a user's viewpoint. There is also major disagreement among experts in providing relevance judgements (Haynes et al. 1990, Hersh and Hickam 1994). Further, these judgements of relevance are affected not only by the expertise of the judge, but also by the order of the documents (Eisenberg and Barry 1988; Schamber et al. 1990). It has been argued that relevance is not fixed, that it varies over time (Meadow 1992). The meaning and the relevance of a document can thus be different for different users and can be inferred only in the context of the user's situation. Relevance, therefore, is subjective, dynamic, and multi-dimensional in nature (Saracevic 1975, Mizzaro 1998, Harter 1992).

Another drawback of the system-driven approach is that it removes the end users from the retrieval process, substituting them with the queries and judgements provided with the test collection. This allows fast experimentation but makes it difficult to evaluate the effect of interactive IR techniques and is suitable only for non-interactive environment (Draper and Dunlop). In an interactive setting, a user normally starts with a query, which goes through many refinements to eventually get the desired documents. The test-collection approach poses a problem in such an environment. As performance of the IR system will eventually be measured in terms of it ability to retrieve documents relevant to a user's query, it seems realistic to follow a user-centered approach to evaluation. Such an approach will result in a much more direct measure of the overall goal. A number of measures have been proposed for interactive IR including relative relevance (RR), ranked half life (RHL), and cumulated gain (CG). Details of these measures can be found in Hersh et al. (1995), Borlund and Ingwerson (1998), and Järvelin and Kekäläinen (2000).

The subjective nature of interactive IR has been highlighted (Borlund and Ingwerson 1997) and attempts have been made to integrate cognitive

Scanned by CamScanner

theory into IR evaluation. However, efforts in this direction have been limited. A task oriented, user-centred, non-interactive evaluation methodology has been proposed by Reid (2000), in which the basic unit of evaluation is task rather than query. More recently, an interactive IR evaluation model has been proposed by Borlund (2000, 2003) to evaluate interactive IR systems. The key elements of an interactive IR model are the use of realistic scenarios (simulated work tasks) and alternative performance measures such as RR and RHL. However, user-centred evaluation methods are expensive both in terms of time and resources. A properly designed user-centred evaluation, with a few exceptions, requires a sufficiently large, representative sample of actual users of retrieval systems. The systems to be compared must be equally well-developed and equipped with the appropriate user interface. The subject must be trained with these systems. Further, it is difficult to develop a standard interactive evaluation methodology that will allow for comparison across different systems and users (Reid 2000). Because of these considerations, recall and precision remain the most popular and standard measure for evaluating IR system performance.

UMMARY

- Information retrieval (IR) deals with the organization, storage, retrieval, and evaluation of information relevant to a user's query.
- An IR system does not return the actual information but returns the documents containing that information.
- The actual text of the document is not used in the retrieval process. Instead, documents in a collection are frequently represented through a set of index terms or keywords.
- The process of transforming document text to some representation of it is known as *indexing*.
- A common lexical processing of index terms involves elimination of stop words.
- An IR model is a pattern that defines several aspects of the retrieval procedure, for example, how the documents and users' queries are represented, how the system retrieves relevant documents according to users' queries, and how retrieved documents are ranked.
- Classical IR models, such as Boolean, vector space, and probabilistic, are based on mathematical knowledge that is easily recognized and well understood.

Scanned by CamScanner

1. What is the difference between data retrieval and information retrieval?
2. How does stemming affect the performance of an IR system?
3. What are the benefits of eliminating stop words. Give examples in which stop word elimination may be harmful.
4. Given the following document:

   The oldest Chinese language we know about is on oracle bones. Priests scratched questions on animal bones and then held the bones in a fire so that they cracked. The places where the cracks crossed the pictograms were thought to give the answers from the god.

   Assume that raw term frequency is used and the stop words are 'the', 'we', 'is', 'on', 'and', 'then', 'in', 'a', 'so', 'that', 'they', 'were', 'to', 'where', 'but', 'only', 'out'.

   Find the vector representation of the above document. Use porter stemmer for stemming.

Scanned by CamScanner

5. In a collection of 10,000 documents, the following words occurs in the following number of documents:

   'oasis' occurs in 400 documents
   'place' occurs in 3,500 documents
   'desert' occurs in 800 documents
   'water' occurs in 800 documents
   'comes' occurs in 800 documents
   'beneath' occurs in 800 documents
   'ground' occurs in 800 documents

   Calculate tf-idf term vector for the following document:

   'An oasis is a place in a desert where water comes out from beneath the ground'.

   Perform stop word removal using the stop word list in exercise 4, and order tokens in the vector alphabetically.

6. Use the stop words given in Exercise 4 to construct vectors (assume simple term frequency weight) for the following documents:

   'An oasis is a place in a desert where water comes out from beneath the ground'.

   'Most people think of desert as a vast sandy region, but only 20% of the world's deserts are sandy'.

   Find cosine similarity between the two vectors.

7. Using Zipf's law, estimate the following in terms of constant *K*.
   (i) Number of distinct terms that have a frequency equal to *f*.
   (ii) Number of distinct terms in the collection.
   (iii) Number of distinct terms that appear only once in the corpus.
8. How well does LSI work? What will happen if k is too big or too small?
9. Define recall and precision. What will happen if there is no relevant document in a collection for a given query? What will the recall–precision curve look like if all the documents in a collection were relevant?
10. A user submitted a query to an IR system. Out of the first 15 documents returned by the system, those ranked 1, 2, 5, 8, and 12 were relevant. Compute non-interpolated average precision for this retrieval. Assume that the total number of relevant documents is six.
11. Interpolate precision for the retrieval situation described in Exercise 10 at the 11- recall points, viz., {0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0} and draw recall–precision curve.

Scanned by CamScanner

CHAPTER 12
LEXICAL RESOURCES

CHAPTER OVERVIEW

This chapter introduces various tools and lexical resources used in text processing applications and provides a ready reference of these. In particular, it introduces the reader with tools such as stemmers and taggers, lexical resources such as WordNet and FrameNet, and test collections (corpuses) that are freely available for research purpose.

12.1 INTRODUCTION

A whole range of tools and lexical resources have been developed to ease the task of researchers working with natural language processing (NLP). Many of these are open sources, i.e., readers can download them off the Internet. This chapter introduces some of the freely available resources. The motivation behind including this chapter comes from the belief that *knowing where the information is, is half of the information.*

We hope that providing a ready reference of what is available where, will save a lot of time and effort, especially for young researchers and those who are new to the field. All the material presented in this chapter is already available, at the links provided with the discussion or in the form of scholarly articles published on that resource. We bring these resources together and offer a brief discussion on them. In particular, we discuss lexical resources such as WordNet and FrameNet, and tools such as stemmers, taggers, and parsers, and freely available test corpuses for various text-processing applications. We begin our discussion with WordNet in Section 12.2. Section 12.3 discusses FrameNet. Stemmers are discussed in Section 12.4. We present a list of available part-of-speech taggers in Section 12.5. The next section presents a list of document collections. And finally, relevant journals and conferences are listed in Section 12.7.

Scanned by CamScanner

12.2 WORDNET

WordNet[1] (Miller 1990, 1995) is a large lexical database for the English language. Inspired by psycholinguistic theories, it was developed and is being maintained at the Cognitive Science Laboratory, Princeton University, under the direction of George A. Miller. WordNet consists of three databases—one for nouns, one for verbs, and one for both adjectives and adverbs. Information is organized into sets of synonymous words called *synsets*, each representing one base concept. The synsets are linked to each other by means of lexical and semantic relations. Lexical relations occur between word-forms (i.e., senses) and semantic relations between word meanings. These relations include synonymy, hypernymy/hyponymy, antonymy, meronymy/holonymy, troponymy, etc. A word may appear in more than one synset and in more than one part-of-speech. The meaning of a word is called sense. WordNet lists all senses of a word, each sense belonging to a different synset. WordNet's sense-entries consist of a set of synonyms and a gloss. A gloss consists of a dictionary-style definition and examples demonstrating the use of a synset in a sentence, as shown in Figure 12.1. The figure shows the entries for

12.1.1 Noun

1. **read** (something that is read) *"the article was a very good read"*

12.1.2 Verb

1. **read** (interpret something that is written or printed) *"read the advertisement"; "Have you read Salman Rushdie?"*
2. **read**, say (have or contain a certain wording or form) *"The passage reads as follows"; "What does the law say?"*
3. **read** (look at, interpret, and say out loud something that is written or printed) *"The King will read the proclamation at noon"*
4. **read**, scan (obtain data from magnetic tapes) *"This dictionary can be read by the computer"*
5. **read** (interpret the significance of, as of palms, tea leaves, intestines, the sky; also of human behaviour) *"She read the sky and predicted rain"; "I can't read his strange behavior"; "The fortune teller read his fate in the crystal ball"*
6. take, **read** (interpret something in a certain way; convey a particular meaning or impression) *"I read this address as a satire"; "How should I take this message?"; "You can't take credit for this!"*
7. learn, study, **read**, take (be a student of a certain subject) *"She is reading for the bar exam"*
8. **read**, register, show, record (indicate a certain reading; of gauges and instruments) *"The thermometer showed thirteen degrees below zero"; "The gauge read `empty'"*
9. **read** (audition for a stage role by reading parts of a role) *"He is auditioning for `Julius Caesar' at Stratford this year"*
10. **read** (to hear and understand) *"I read you loud and clear!"*
11. understand, **read**, interpret, translate (make sense of a language) *"She understands French"; "Can you read Greek?"*

Figure 12.1 WordNet 2.0 entry for 'read'

[1] http://wordnet.princeton.edu/

Scanned by CamScanner

the word 'read'. 'Read' has one sense as a noun and 11 senses as a verb. Glosses help differentiate meanings. Figures 12.2, 12.3, and 12.4 show some of the relationships that hold between nouns, verbs, and adjectives and adverbs. Nouns and verbs are organized into hierarchies based on the hypernymy/hyponymy relation, whereas adjectives are organized into clusters based on antonym pairs (or triplets). Figure 12.5 shows a hypernym chain for 'river' extracted from WordNet. Figure 12.6 shows the troponym relations for the verb 'laugh'.

Relation	*Definition*	*Example*
Hypernym	From concepts to super-ordinates	oak→ tree
Hyponym	From concepts to subtypes	oak→ white oak
Meronym	From wholes to parts	tree → trunk
Holonym	From parts to wholes	trunk → tree
Antonym	Opposites	victory → defeat

Figure 12.2 Noun relations in WordNet

Relation	*Definition*	*Example*
Hypernym	From events to super-ordinate events	wander→ travel
Troponym	From events to their subtypes	walk → stroll
Entails	From events to the events they entail	snore → sleep
Antonym	Opposites	increase → decrease

Figure 12.3 Verb relations in WordNet

Relation	*Definition*	*Example*
Antonym (adjective)	Opposite	heavy → light
Antonym (adverb)	Opposite	quickly → slowly

Figure 12.4 Adjective and adverb relations in WordNet

1 sense of 'river'
Sense 1
river — (a large natural stream of water (larger than a creek); 'the river was navigable for 50 miles')
=> stream, watercourse — (a natural body of running water flowing on or under the earth)
=> body of water, water — (the part of the earth's surface covered with water (such as a river or lake or ocean); 'they invaded our territorial waters'; 'they were sitting by the water's edge')
=> thing — (a separate and self-contained entity)
=> entity — (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

Figure 12.5 Hypernym chain for 'river'

Scanned by CamScanner

WordNet is freely and publicly available for download from *http://wordnet.princeton.edu/obtain.*

WordNets for other languages have also been developed, e.g., EuroWordNet and Hindi WordNet. EuroWordNet covers European languages, including English, Dutch, Spanish, Italian, German, French, Czech, and Estonian. Other than language internal relations, it also contains multilingual relations from each WordNet to English meanings.

Hindi WordNet has been developed by CFILT (Resource Center for Indian Language Technology Solutions), IIT Bombay.[2] Its database consists of more than 26,208 synsets and 56,928 Hindi words.[3] It is organized using the same principles as English WordNet but includes some Hindi specific relations (e.g., causative relations). A total of 16 relations have been used in Hindi WordNet. Each entry consists of synset, gloss, and position of synset in ontology. Figure 12.7 shows the Hindi WordNet entry for the word 'आकांक्षा' (*aakanksha*).

Sense 1
laugh, express joy, express mirth — (produce laughter)
=> bray — (laugh loudly and harshly)
=> bellylaugh — (laugh a deep, hearty laugh)
=> roar, howl — (laugh unrestrainedly and heartily)
=> snicker, snigger — (laugh quietly)
=> giggle, titter — (laugh nervously; 'The girls giggled when the rock star came into the classroom')
=> break up, crack up — (laugh unrestrainedly)
=> cackle — (emit a loud, unpleasant kind of laughing)
=> guffaw, laugh loudly — (laugh boisterously)
=> chuckle, chortle, laugh softly — (laugh quietly or with restraint)
=> convulse — (be overcome with laughter)
=> cachinnate — (laugh loudly and in an unrestrained way)

Figure 12.6 Troponym relation for the word 'laugh'

Hindi WordNet can be obtained from the URL http://*www.cfilt.iitb.ac.in/wordnet/webhwn/.*

CFLIT has also developed a Marathi WordNet. Figure 12.8 shows the Marathi WordNet (*http://www.cfilt.iitb.ac.in/wordnet/webmwn/wn.php*) entry for the word 'पाव' (*pau*).

12.2.1 Applications of WordNet

WordNet has found numerous applications in problems related with IR and NLP. Some of these are discussed here.

[2] www.cfilt.iitb.ac.in/

Scanned by CamScanner

Concept Identification in Natural Language

WordNet can be used to identify concepts pertaining to a term, to suit them to the full semantic richness and complexity of a given information need.

Word Sense Disambiguation

WordNet combines features of a number of the other resources commonly used in disambiguation work. It offers sense definitions of words, identifies synstes of synonyms, defines a number of semantic relations and is freely available. This makes it the (currently) best known and most utilized resource for word sense disambiguation. One of the earliest attempts to use WordNet for word sense disambiguation was in IR by Voorheese (1993). She used WordNet noun hierarchy (hypernym / hyponym) to achieve disambiguation. A number of other researchers have also used WordNet for the same purpose (Resnik 1995, 1997; Sussna 1993).

1. (R) **अपेक्षा, आकांक्षा, अन्वपेक्षा** – किसी पर भरोसा रखने की क्रिया कि अमुक कार्य उसके द्वारा हो जायेगा " हर पिता की अपने पुत्र से यह अपेक्षा रहती है कि वह अपने जीवन में सफल हो "
2. (R) **इच्छा, अभिलाषा, आकांक्षा, ख्वाहिश, आरज़ू, तमन्ना, कामना, तलब, चेष्टा, हसरत, मुराद, पिपासा, प्यास, तृष्णा, मनोकामना, मनोकामना, मनोवांछा, मनोरथ, मनोभावना, मरजी, रजा, मर्जी, मन, रजा, मंशा, लिप्सा, लालसा, तृषा, चाह, अरमान, क्षुधा, भूख, भूक, छुधा, हवस, स्पृहा, अभीप्सा, अनु, अपेक्षिता, अभिकांक्षा, अभिकाम, वांछा, वाञ्छा, वाञ्छा, अभिध्या, अभिलाष, अभिप्रीति, अभिमत, अभिमतता, अभिमति, अभिलास, अभिलासा, अभिलाखा, अभिलाख, अभिलाखना** — वह मनोवृत्ति जो किसी बात या वस्तु की प्राप्ति की ओर ध्यान ले जाती है " इंसान की हर इच्छा पूरी नही होती / उसकी ज्ञान पिपासा बढती जा रही है / मेरा आज खाने का मन नहीं है "

Figure 12.7 WordNet entry for the Hindi word आकांक्षा (*aakanksha*)

1. (R) पाव एक चतुर्थांश-चौथा भाग "मी बाजारातून एक पाव चुरमुरे आणले"
2. (R) पाव, पावरोटी-गव्हाचे पीठ आंबवून केलेल एक खाद्यविशेष "मुंबईत बरेच लोक पाव खाऊन गुजराण करतात"
3. (R) पाव-पावाचे वजन "दुकानदार चहाच्या पूडीचे वजन करण्यासाठी पाव शोधत आहे"
4. (R) पाव-एक चतुर्थांश बाटली दारू "एक पाव प्यायल्यावर तो वटवट करू लागलाइ"

Figure 12.8 WordNet entry for the Marathi word पाव (*pau*)

Automatic Query Expansion

WordNet semantic relations can be used to expand queries so that the search for a document is not confined to the pattern-matching of query terms, but also covers synonyms. The work performed by Voorhees (1994) is based on the use of WordNet relations, such as synonyms, hypernyms, and hyponyms, to expand queries.

Scanned by CamScanner

Document Structuring and Categorization

The semantic information extracted from WordNet, and WordNet conceptual representation of knowledge, have been used for text categorization (Scott and Matwin 1998).

Document Summarization

WordNet has found useful application in text summarization. The approach presented by Barzilay and Elhadad (1997) utilizes information from WordNet to compute lexical chains.

12.3 FRAMENET

FrameNet[4] is a large database of semantically annotated English sentences. It is based on principles of frame semantics. It defines a tagset of semantic roles called the frame element. Sentences from the British National Corpus are tagged with these frame elements. The basic philosophy involved is that each word evokes a particular situation with particular participants. FrameNet aims at capturing these situations through case-frame representation of words (verbs, adjectives, and nouns). The word that invokes a frame is called *target word* or *predicate*, and the participant entities are defined using semantic roles, which are called *frame elements*. The FrameNet ontology can be viewed as a semantic level representation of predicate argument structure.

Each frame contains a main lexical item as predicate and associated frame-specific semantic roles, such as AUTHORITIES, TIME, and SUSPECT in the ARREST frame, called frame elements. As an example, consider sentence (12.1) annotated with the semantic roles AUTHORITIES and SUSPECT. The target word in sentence (12.1) is 'nab' which is a verb in the ARREST frame.

[Authorities The police] nabbed [suspect the snatcher]. (12.1)

A COMMUNICATON frame has the semantic roles ADDRESSEE, COMMUNICATOR, TOPIC, and MEDIUM. Figure 12.9 shows the core and non-core frame elements of the COMMUNICATION frame, along with other details. A JUDGEMENT frame contains roles such as JUDGE, EVALUEE, and REASON. A frame may inherit roles from another frame. For example, a STATEMENT frame may inherit from a COMMUNICATION frame; it contains roles such as SPEAKER, ADDRESSEE, and MESSAGE. The following sentences show some of these roles:

[Judge She] [Evaluee blames the police] [Reason for failing to provide enough protection]. (12.2)

[Speaker She] told [Addressee me] [Message 'I'll return by 7:00 pm today']. (12.3)

[4] http://framenet.icsi.berkeley.edu/

Scanned by CamScanner

12.3.1 FrameNet Applications

Gildea and Jurafsky (2002) and Kwon et al. (2004) used FrameNet data for automatic semantic parsing. The shallow semantic role obtained from FrameNet can play an important role in information extraction. However, though the semantic role is same, the syntactic role is different. In sentence (12.4), the word 'match' is the object, while it is the subject in sentence (12.5).

The umpire stopped the match. (12.4)

The match stopped due to bad weather. (12.5)

Semantic roles may help in the question-answering system. For example, the verb 'send' and 'receive' would share the semantic roles SENDER, RECIPIENT, GOODS, etc., (Gildea and Jurafsky 2002) when defined with respect to a common TRANSFER frame. Such common frames allow a question-answering system to answer a question such as 'Who sent packet to Khushbu?' using sentence (12.6).

Khushbu received a packet from the examination cell. (12.6)

Other applications include IR (Mohit and Narayanan 2003), interlingua for machine translation, text summarization, and word sense disambiguation.

Communication

Frame Elements

Core:

Addressee [Add]	Receiver of Message from the Communicator.
Communicator [Com]	The person conveying (written or spoken) a message to another person.
Message [Msg]	A proposition or set of propositions that the Communicator wants the Addressee to convey
Topic [Top]	The entity that the proposition(s) are about.

Non-core:

Amount_of_information [Amo]	The amount of information exchanged when communication occurs.
Depictive [Dep-Act]	The Depictive describes the state of the Communicator.
Duration []	The length of time during which the communication takes place.
Manner [Manr]	The Manner in which the Communicator communicates.
Means [Mns]	The Means by which the Communicator communicates.
Medium [Medium]	The physical or abstract setting in which the Message is conveyed.
Time []	The time at which the communication takes place.

Inherits From:
Is Inherited By: Communication_noise, Statement
Subframe of:
Has Subframes:
Uses: Topic
Is Used By: Claim_ownership, Communication_response, Contacting, Deny_permission, Discussion, Hear, Questioning, Reasoning, Reporting, Request, etc
Is Inchoative of:
Is Causative of:
See Also:

Sample Predicates

communicate, indicate, signal, speech

Figure 12.9 Frame elements of communication frame

Scanned by CamScanner

12.4 STEMMERS

As discussed in Chapter 3, stemming, often called conflation, is the process of reducing inflected (or sometimes derived) words to their base or root form. The stem need not be identical to the morphological base of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Stemming is useful in search engines for query expansion or indexing and other NLP problems. Stemming programs are commonly referred to as stemmers. The most common algorithm for stemming English is Porter's algorithm[5] (Porter 1980). Other existing stemmers include Lovins[6] stemmer (Lovins 1968) and a more recent one called the Paice/Husk stemmer[7] (Paice 1990). Figure 12.10 shows a sample text and output produced using these stemmers.

Input Text:
Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation.

Output:
Lovins stemmer: such an analys can rev feature that ar not eas vis from the vari in th individu gen and can lead to a picture of expres that is mor biolog transpar and acces to interpres
Porter's stemmer: such an analysi can reveal feature that ar not easily visible from the variat in the individu gene and can lead to a picture of express that is more biolog transpar and access to interpret
Paice stemmer: Such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Figure 12.10 Stemmed text using different stemmers

12.4.1 Stemmers for European Languages

There are many stemmers available for English and other languages. Snowball[8] presents stemmers for English, Russian, and a number of other European languages, including French, Spanish, Portuguese, Hungarian, Italian, German, Dutch, Swedish, Norwegian, Danish, and Finnish. The links for stemming algorithms for these languages can be found at *http://snowball.tartarus.org/texts/stemmersoverview.html.*

[5] http://tartarus.org/~martin/PorterStemmer/
[6] http://sourceforge.net/project/showfiles.php?group_id=24260
[7] http://www.comp.lancs.ac.uk/computing/research/stemming/Links/implementations.htm

Scanned by CamScanner

12.4.2 Stemmers for Indian Languages

Standard stemmers are not yet available for Hindi and other Indian languages. The major research on Hindi stemming has been accomplished by Ramanathan and Rao (2003) and Majumder et al. (2007). Ramanathan and Rao (2003) based their work on the use of handcrafted suffix lists. Majumder et al. (2007) used a cluster-based approach to find classes of root words and their morphological variants. They used a task-based evaluation of their approach and reported that stemming improves recall for Indian languages. Their observation on Indian languages was based on a Bengali data set. The Resource Centre of Indian Language Technology (CFILT), IIT Bombay has also developed stemmers for Indian languages, which are available at *http://www.cfilt.iitb.ac.in.*

12.4.3 Stemming Applications

Stemmers are common elements in search and retrieval systems such as Web search engines. Stemming reduces the variants of a word to same stem. This reduces the size of the index and also helps retrieve documents that contain variants of a query terms. For example, a user issuing a query for documents on 'astronauts' would like documents on 'astronaut' as well. Stemming permits this by reducing both versions of the word to the same stem. However, the effectiveness of stemming for English query systems is not too great, and in some cases may even reduce precision.

Text summarization and text categorization also involve term frequency analysis to find features. In this analysis, stemming is used to transform various morphological forms of words into their stems.

12.5 PART-OF-SPEECH TAGGER

Part-of-speech tagging is used at an early stage of text processing in many NLP applications such as speech synthesis, machine translation, IR, and information extraction. In IR, part-of-speech tagging can be used in indexing (for identifying useful tokens like nouns), extracting phrases and for disambiguating word senses. The rest of this section presents a number of part-of-speech taggers that are already in place.

12.5.1 Stanford Log-linear Part-of-Speech (POS) Tagger

This POS Tagger is based on maximum entropy Markov models. The key features of the tagger are as follows:

(i) It makes explicit use of both the preceding and following tag contexts via a dependency network representation.

Scanned by CamScanner

(ii) It uses a broad range of lexical features.
(iii) It utilizes priors in conditional log-linear models.

The reported accuracy of this tagger on the Penn Treebank WSJ is 97.24%, which amounts to an error reduction of 4.4% on the best previous single automatically learned tagging result (Tuotanova et al. 2003). Details on the tagger can be found at the link *http://nlp.stanford.edu/software/tagger.shtml.*

12.5.2 A Part-of-Speech Tagger for English[9]

This tagger uses a bi-directional inference algorithm for part-of-speech tagging. It is based on maximum entropy Markov models (MEMM). The algorithm can enumerate all possible decomposition structures and find the highest probability sequence together with the corresponding decomposition structure in polynomial time. Experimental results of this part-of-speech tagger show that the proposed bi-directional inference methods consistently outperform unidirectional inference methods and bi-directional MEMMs give comparable performance to that achieved by state-of-the-art learning algorithms, including kernel support vector machines (Tsuruoka and Tsujii 2005).

12.5.3 TnT tagger[10]

Trigrams'n'Tags or TnT (Brants 2000) is an efficient statistical part-of-speech tagger. This tagger is based on hidden Markov models (HMM) and uses some optimization techniques for smoothing and handling unknown words. It performs at least as well as other current approaches, including the maximum entropy framework. Table 12.1 shows tagged text of document #93 of the CACM collection.

Table 12.1 Doc #93 of CACM collection tagged using TnT tagger

A	DT	simple	JJ
technique	NN	algebraic	JJ
is	VBZ	formulas	NNS
shown	VBN	into	IN
for	IN	a	DT
enabling	VBG	three	CD
a	DT	address	NN
computer	NN	computer	NN
to	TO	code	NN
translate	VB		

[9] www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger/
[10] www.coli.uni-saarland.de/~thorsten/tnt/

Scanned by CamScanner

12.5.4 Brill Tagger

Brill (1992) described a trainable rule-based tagger that obtained performance comparable to that of stochastic taggers. It uses transformation-based learning to automatically induce rules. A number of extensions to this rule-based tagger have been proposed by Brill (1994). He describes a method for expressing lexical relations in tagging that stochastic taggers are currently unable to express. It implements a rule-based approach to tagging unknown words. It demonstrates how the tagger can be extended into a k-best tagger, where multiple tags can be assigned to words in some cases of uncertainty. Brill tagger is available for download at the link *http://www.cs.jhu.edu/~brill/RBT1_14.tar.Z.*

12.5.5 CLAWS Part-of-Speech Tagger for English

Constituent likelihood automatic word-tagging system (CLAWS) is one of the earliest probabilistic taggers for English. It was developed at the University of Lancaster (*http://ucrel.lancs.ac.uk/claws*). The latest version of the tagger, CLAWS4, can be considered a hybrid tagger as it involves both probabilistic and rule-based elements. It has been designed so that it can be easily adapted to different types of text in different input formats. CLAWS has achieved 96–97% accuracy. The precise degree of accuracy varies according to the type of text. For more information on the CLAWS tagger, see Garside (1987), Leech, Garside, and Bryant (1994), Garside (1996), and Garside and Smith (1997).

12.5.6 Tree-Tagger

Tree-Tagger (Schmidt 1994) is a probabilistic tagging method. It avoids problems faced by the Markov model methods when estimating transition probabilities from sparse data, by using a decision tree to estimate transition probabilities. The decision tree automatically determines the appropriate size of the context to be used in estimation. The reported accuracy for the tagger is above 96% on the Penn-Treebank WSJ corpus. The tagger is available at the link *http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html.*

12.5.7 ACOPOST: A Collection of POS Taggers[11]

ACOPOST is a set of freely available POS taggers. The taggers in the set are based on different frameworks. The programs are written in C. ACOPOST currently consists of the following four taggers.

Scanned by CamScanner

Maximum Entropy Tagger (MET)

This tagger is based on a framework suggested by Ratnaparkhi (1997). It uses an iterative procedure to successively improve parameters for a set of features that help to distinguish between relevant contexts.

Trigram Tagger (T3)

This tagger is based on HMM. The states in the model are tag pairs that emit words. The technique has been suggested by Rabiner (1990) and the implementation is influenced by Brants (2000).

Error-driven Transformation-based Tagger (TBT)

This tagger is based on the transformation-based tagging approach proposed by Brill (1993). It uses annotated corpuses to learn transformation rules, which are then used to change the assigned tag using contextual information.

Example-based Tagger (ET)

The underlying assumption of example-based models (also called memory-based, instance-based or distance-based models) is that cognitive behaviour can be achieved by looking at past experiences that match the current problem, instead of learning and applying abstract rules. This framework has been suggested for NLP by Daelemans et al. (1996).

12.5.7 POS Tagger for Indian Languages

The automatic text processing of Hindi and other Indian languages is constrained heavily due to lack of basic tools and large annotated corpuses. Research groups are now focusing on removing these bottlenecks. The work on the development of tools, techniques, and corpora is going on at several places such as CDAC, IIT Bombay, IIIT Hyderabad, University of Hyderabad, CIIL Mysore, and University of Lancaster. IIT Bombay is involved in the development of morphology analysers and part-of-speech taggers for Hindi and Marathi. Both these languages have rich morphological structures. Their approach is based on *bootstrapping on a small corpus tagged* by a rule-based tagger and then applying statistical techniques to train a machine. More information can be found at *http://ltrc.iiit.net* and *www.cse.iitb.ac.in*. Work on Urdu part-of-speech taggers has been reported by Hardie (2003) and Baker et al. (2004).

Scanned by CamScanner

Research corpora have been developed for a number of NLP-related tasks. In the following section, we point out few of the available standard document collections for a variety of NLP-related tasks, along with their Internet links.

12.6.1 IR Test Collection

We have already provided a list of IR test document collection in Chapter 9. Glasgow University, UK, maintains a list of freely available IR test collections. Table 12.2 lists the sources of those and few more IR test collections.

LETOR (learning to rank) is a package of benchmark data sets released by Microsoft Research Asia. It consists of two datasets OHSUMED and TREC (TD2003 and TD2004). LETOR is packaged with extracted features for each query-document pair in the collection, baseline results of several state-of-the-art learning-to-rank algorithms on the data and evaluation tools. The data set is aimed at supporting future research in the area of learning ranking function for information retrieval.

Table 12.2 IR test collection

LETOR	*http://research.microsoft.com/users/tyliu/LETOR/*
LISA	
CACM	
CISI	
MEDLINE	*http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/*
Cranfield	
TIME	
ADI	

12.6.2 Summarization Data

Evaluating a text summarizing system requires existence of 'gold summaries'. DUC provides document collections with known extracts and abstracts, which are used for evaluating performance of summarization systems submitted at TREC conferences. Figure12.11 shows a sample document and its extract from DUC 2002 summarization data.

Scanned by CamScanner

<DOC>
<DOCNO> AP880911-0016 </DOCNO>
<FILEID>AP-NR-09-11-88 0423EDT</FILEID>
<FIRST>r i BC-HurricaneGilbert 09-11 0339</FIRST>
<SECOND>BC-Hurricane Gilbert,0348</SECOND>
<HEAD>Hurricane Gilbert Heads Toward Dominican Coast</HEAD>
<BYLINE>By RUDDY GONZALEZ</BYLINE>
<BYLINE>Associated Press Writer</BYLINE>
<DATELINE>SANTO DOMINGO, Dominican Republic (AP) </DATELINE>
<TEXT>

Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.

The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.

"There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday.

Cabral said residents of the province of Barahona should closely follow Gilbert's movement. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.

Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.

The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.

The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.

Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet to Puerto Rico's south coast. There were no reports of casualties.

San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.

On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast. Residents returned home, happy to find little damage from 80 mph winds and sheets of rain.

Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane. The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.

</TEXT>
</DOC>

Extract

Tropical Storm Gilbert in the eastern Caribbean strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday to be about 140 miles south of Puerto Rico and 200 miles southeast of Santo Domingo. It is moving westward at 15mph with a broad area of cloudiness and heavy weather with sustained winds of 75mph gusting to 92mph. The Dominican Republic's Civil Defense alerted that country's heavily populated south coast and the National Weather Service in San Juan, Puerto Rico issued a flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.

Figure 12.11 Sample document from DUC 2002 and its extract

Scanned by CamScanner

12.6.3 Word Sense Disambiguation

SEMCOR[12] is a sense-tagged corpus used in disambiguation. It is a subset of the Brown corpus, sense-tagged with WordNet synsets. Open Mind Word Expert[13] attempts to create a very large sense-tagged corpus. It collects word sense tagging from the general public over the Web.

12.6.4 Asian Language Corpora

The multilingual EMILLE corpus is the result of the enabling minority language engineering (EMILLE) project at Lancaster University, UK. The project focuses on generation of data, software resources and basic language engineering tools for the NLP of south Asian languages. Central Institute for Indian Languages (CIIL), the Indian partner in the project, extended the set of target languages to include a number of Indian languages. CIIL provides a wider range of data in these languages from a wide range of genres. The data sources that EMILLE made available include monolingual written and spoken corpuses, parallel and annotated corpuses. The full EMILLE/CIIL corpus is available for free, but for research use only, at the link *http://www.elda.org/catalogue/en/text/W0037.html.* Further details about the corpus can be found in the manual at the site *http://www.emille.lancs.ac.uk/manual.pdf.*

Corpus building in these languages is constrained by the scarcity of repositories of electronic text. The monolingual corpus includes written data for 14 South Asian languages and spoken data for five languages (Hindi, Bengali, Gujrati, Punjabi, and Urdu). The spoken corpus was constructed from radio broadcasts on the BBC Asia network. The parallel corpus contains English text and its translation in five languages. The text includes UK government advice leaflets which are published in multiple languages. The corpus is aligned at sentence level. The parallel corpus provided by EMILLE corpus is a valuable resource for statistical machine translation research. The annotated component includes Urdu data annotated for part-of-speech tagging, and a Hindi corpus annotated to show nature of demonstrative use.

Scanned by CamScanner