Let k_i be an index term, d_j be a document, and $w_{i,j} \geq 0$ be a *weight* associated with the pair (k_i, d_j). This weight quantifies the importance of the index term for describing the document semantic contents.

Definition *Let t be the number of index terms in the system and k_i be a generic index term. $K = \{k_1, \ldots, k_t\}$ is the set of all index terms. A weight $w_{i,j} > 0$ is associated with each index term k_i of a document d_j. For an index term which does* not *appear in the document text, $w_{i,j} = 0$. With the document d_j is associated an index term vector $\vec{d_j}$ represented by $\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$. Further, let g_i be a function that returns the weight associated with the index term k_i in any t-dimensional vector (i.e., $g_i(\vec{d_j}) = w_{i,j}$).*

As we later discuss, the index term weights are usually assumed to be mutually independent. This means that knowing the weight $w_{i,j}$ associated with the pair (k_i, d_j) tells us nothing about the weight $w_{i+1,j}$ associated with the pair (k_{i+1}, d_j). This is clearly a simplification because occurrences of index terms in a document are not uncorrelated. Consider, for instance, that the terms *computer* and *network* are used to index a given document which covers the area of computer networks. Frequently, in this document, the appearance of one of these two words attracts the appearance of the other. Thus, these two words are correlated and their weights could reflect this correlation. While mutual independence seems to be a strong simplification, it does simplify the task of computing index term weights and allows for fast ranking computation. Furthermore, taking advantage of index term correlations for improving the final document ranking is not a simple task. In fact, none of the many approaches proposed in the past has clearly demonstrated that index term correlations are advantageous (for ranking purposes) with general collections. Therefore, unless clearly stated otherwise, we assume mutual independence among index terms. In Chapter 5 we discuss modern retrieval techniques which are based on term correlations and which have been tested successfully with particular collections. These successes seem to be slowly shifting the current understanding towards a more favorable view of the usefulness of term correlations for information retrieval systems.

The above definitions provide support for discussing the three classic information retrieval models, namely, the Boolean, the vector, and the probabilistic models, as we now do.

2.5.2 Boolean Model

The Boolean model is a simple retrieval model based on set theory and Boolean algebra. Since the concept of a set is quite intuitive, the Boolean model provides a framework which is easy to grasp by a common user of an IR system. Furthermore, the queries are specified as Boolean expressions which have precise semantics. Given its inherent simplicity and neat formalism, the Boolean model received great attention in past years and was adopted by many of the early commercial bibliographic systems.

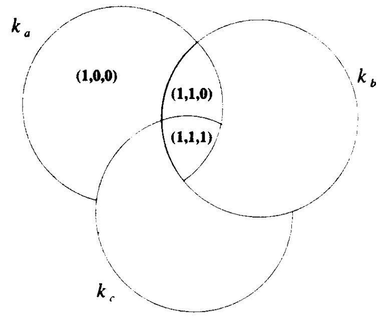

Figure 2.3 The three conjunctive components for the query $[q = k_a \wedge (k_b \vee \neg k_c)]$.

Unfortunately, the Boolean model suffers from major drawbacks. First, its retrieval strategy is based on a binary decision criterion (i.e., a document is predicted to be either relevant or non-relevant) without any notion of a grading scale, which prevents good retrieval performance. Thus, the Boolean model is in reality much more a data (instead of information) retrieval model. Second, while Boolean expressions have precise semantics, frequently it is not simple to translate an information need into a Boolean expression. In fact, most users find it difficult and awkward to express their query requests in terms of Boolean expressions. The Boolean expressions actually formulated by users often are quite simple (see Chapter 10 for a more thorough discussion on this issue). Despite these drawbacks, the Boolean model is still the dominant model with commercial document database systems and provides a good starting point for those new to the field.

The Boolean model considers that index terms are present or absent in a document. As a result, the index term weights are assumed to be all binary, i.e., $w_{i,j} \in \{0,1\}$. A query q is composed of index terms linked by three connectives: *not*, *and*, *or*.Thus, a query is essentially a conventional Boolean expression which can be represented as a disjunction of conjunctive vectors (i.e., in *disjunctive normal form* — DNF). For instance, the query $[q = k_a \wedge (k_b \vee \neg k_c)]$ can be written in disjunctive normal form as $[\vec{q}_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)]$, where each of the components is a binary weighted vector associated with the tuple (k_a, k_b, k_c). These binary weighted vectors are called the conjunctive components of $\vec{q}_{dnf}$. Figure 2.3 illustrates the three conjunctive components for the query q.

Definition *For the Boolean model, the index term weight variables are all binary i.e., $w_{i,j} \in \{0,1\}$. A query q is a conventional Boolean expression. Let $\vec{q}_{dnf}$ be the disjunctive normal form for the query q. Further, let $\vec{q}_{cc}$ be any of the conjunctive components of $\vec{q}_{dnf}$. The similarity of a document d_j to the query q is defined as*

$$sim(d_j, q) = \begin{cases} 1 & if\ \exists \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall_{k_i},\ g_i(\vec{d_j}) = g_i(\vec{q}_{cc})) \\ 0 & otherwise \end{cases}$$

If $sim(d_j, q) = 1$ then the Boolean model predicts that the document d_j is relevant to the query q (it might not be). Otherwise, the prediction is that the document is not *relevant.*

The Boolean model predicts that each document is either *relevant* or *non-relevant*. There is no notion of a *partial match* to the query conditions. For instance, let d_j be a document for which $\vec{d_j} = (0, 1, 0)$. Document d_j includes the index term k_b but is considered non-relevant to the query $[q = k_a \wedge (k_b \vee \neg k_c)]$.

The main *advantages* of the Boolean model are the clean formalism behind the model and its simplicity. The main *disadvantages* are that exact matching may lead to retrieval of too few or too many documents (see Chapter 10). Today, it is well known that index term weighting can lead to a substantial improvement in retrieval performance. Index term weighting brings us to the vector model.

2.5.3 Vector Model

The vector model [697, 695] recognizes that the use of binary weights is too limiting and proposes a framework in which partial matching is possible. This is accomplished by assigning *non-binary* weights to index terms in queries and in documents. These term weights are ultimately used to compute the *degree of similarity* between each document stored in the system and the user query. By sorting the retrieved documents in decreasing order of this degree of similarity, the vector model takes into consideration documents which match the query terms only partially. The main resultant effect is that the ranked document answer set is a lot more precise (in the sense that it better matches the user information need) than the document answer set retrieved by the Boolean model.

Definition *For the vector model, the weight $w_{i,j}$ associated with a pair (k_i, d_j) is positive and non-binary. Further, the index terms in the query are also weighted. Let $w_{i,q}$ be the weight associated with the pair $[k_i, q]$, where $w_{i,q} \geq 0$. Then, the query vector $\vec{q}$ is defined as $\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$ where t is the total number of index terms in the system. As before, the vector for a document d_j is represented by $\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$.*

Therefore, a document d_j and a user query q are represented as t-dimensional vectors as shown in Figure 2.4. The vector model proposes to evaluate the degree of similarity of the document d_j with regard to the query q as the correlation between the vectors $\vec{d_j}$ and $\vec{q}$. This correlation can be quantified, for instance, by the *cosine of the angle* between these two vectors. That is,

$$\begin{aligned} sim(d_j, q) &= \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} \\ &= \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}} \end{aligned}$$

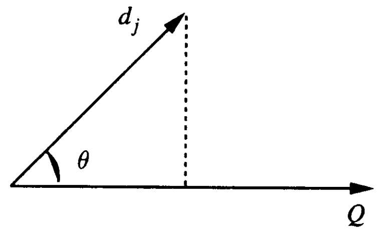

Figure 2.4 The cosine of θ is adopted as $sim(d_j, q)$.

where $|\vec{d_j}|$ and $|\vec{q}|$ are the norms of the document and query vectors. The factor $|\vec{q}|$ does not affect the ranking (i.e., the ordering of the documents) because it is the same for all documents. The factor $|\vec{d_j}|$ provides a normalization in the space of the documents.

Since $w_{i,j} \geq 0$ and $w_{i,q} \geq 0$, $sim(q, d_j)$ varies from 0 to +1. Thus, instead of attempting to predict whether a document is relevant or not, the vector model ranks the documents according to their *degree of similarity* to the query. A document might be retrieved even if it matches the query only *partially*. For instance, one can establish a threshold on $sim(d_j, q)$ and retrieve the documents with a degree of similarity above that threshold. But to compute rankings we need first to specify how index term weights are obtained.

Index term weights can be calculated in many different ways. The work by Salton and McGill [698] reviews various term-weighting techniques. Here, we do not discuss them in detail. Instead, we concentrate on elucidating the main idea behind the most effective term-weighting techniques. This idea is related to the basic principles which support clustering techniques, as follows.

Given a collection C of objects and a *vague* description of a set A, the goal of a simple clustering algorithm might be to separate the collection C of objects into two sets: a first one which is composed of objects related to the set A and a second one which is composed of objects not related to the set A. Vague description here means that we do not have complete information for deciding precisely which objects are and which are not in the set A. For instance, one might be looking for a set A of cars which have a price *comparable* to that of a Lexus 400. Since it is not clear what the term *comparable* means exactly, there is not a precise (and unique) description of the set A. More sophisticated clustering algorithms might attempt to separate the objects of a collection into various clusters (or classes) according to their properties. For instance, patients of a doctor specializing in cancer could be classified into five classes: terminal, advanced, metastasis, diagnosed, and healthy. Again, the possible class descriptions might be imprecise (and not unique) and the problem is one of deciding to which of these classes a new patient should be assigned. In what follows, however, we only discuss the simpler version of the clustering problem (i.e., the one which considers only two classes) because all that is required is a decision on which documents are predicted to be relevant and which ones are predicted to be not relevant (with regard to a given user query).

To view the IR problem as one of clustering, we refer to the early work of Salton. We think of the documents as a collection C of objects and think of the user query as a (vague) specification of a set A of objects. In this scenario, the IR problem can be reduced to the problem of determining which documents are in the set A and which ones are not (i.e., the IR problem can be viewed as a clustering problem). In a clustering problem, two main issues have to be resolved. First, one needs to determine what are the features which better describe the objects in the set A. Second, one needs to determine what are the features which better distinguish the objects in the set A from the remaining objects in the collection C. The first set of features provides for quantification of *intra-cluster* similarity, while the second set of features provides for quantification of *inter-cluster* dissimilarity. The most successful clustering algorithms try to balance these two effects.

In the vector model, intra-clustering similarity is quantified by measuring the raw frequency of a term k_i inside a document d_j. Such term frequency is usually referred to as the *tf factor* and provides one measure of how well that term describes the document contents (i.e., intra-document characterization). Furthermore, inter-cluster dissimilarity is quantified by measuring the inverse of the frequency of a term k_i among the documents in the collection. This factor is usually referred to as the *inverse document frequency* or the *idf factor*. The motivation for usage of an idf factor is that terms which appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one. As with good clustering algorithms, the most effective term-weighting schemes for IR try to balance these two effects.

Definition *Let N be the total number of documents in the system and n_i be the number of documents in which the index term k_i appears. Let $freq_{i,j}$ be the raw frequency of term k_i in the document d_j (i.e., the number of times the term k_i is mentioned in the text of the document d_j). Then, the normalized frequency $f_{i,j}$ of term k_i in document d_j is given by*

$$f_{i,j} = \frac{freq_{i,j}}{max_l \ freq_{l,j}} \tag{2.1}$$

where the maximum is computed over all terms which are mentioned in the text of the document d_j. If the term k_i does not appear in the document d_j then $f_{i,j} = 0$. Further, let idf_i, inverse document frequency for k_i, be given by

$$idf_i = \log \frac{N}{n_i} \tag{2.2}$$

The best known term-weighting schemes use weights which are given by

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i} \tag{2.3}$$

or by a variation of this formula. Such term-weighting strategies are called tf-idf schemes.

Several variations of the above expression for the weight $w_{i,j}$ are described in an interesting paper by Salton and Buckley which appeared in 1988 [696]. However, in general, the above expression should provide a good weighting scheme for many collections.

For the query term weights, Salton and Buckley suggest

$$w_{i,q} = \left(0.5 + \frac{0.5\ freq_{i,q}}{max_l\ \ freq_{l,q}}\right) \times \log \frac{N}{n_i} \tag{2.4}$$

where $freq_{i,q}$ is the raw frequency of the term k_i in the text of the information request q.

The main *advantages* of the vector model are: (1) its term-weighting scheme improves retrieval performance; (2) its partial matching strategy allows retrieval of documents that *approximate* the query conditions; and (3) its cosine ranking formula sorts the documents according to their degree of similarity to the query. Theoretically, the vector model has the *disadvantage* that index terms are assumed to be mutually independent (equation 2.3 does *not* account for index term dependencies). However, in practice, consideration of term dependencies might be a disadvantage. Due to the locality of many term dependencies, their indiscriminate application to all the documents in the collection might in fact *hurt* the overall performance.

Despite its simplicity, the vector model is a resilient ranking strategy with general collections. It yields ranked answer sets which are difficult to improve upon without query expansion or relevance feedback (see Chapter 5) within the framework of the vector model. A large variety of alternative ranking methods have been compared to the vector model but the consensus seems to be that, in general, the vector model is either superior or almost as good as the known alternatives. Furthermore, it is simple and fast. For these reasons, the vector model is a popular retrieval model nowadays.

2.5.4 Probabilistic Model

In this section, we describe the classic probabilistic model introduced in 1976 by Roberston and Sparck Jones [677] which later became known as the *binary independence retrieval* (BIR) model. Our discussion is intentionally brief and focuses mainly on highlighting the key features of the model. With this purpose in mind, we do not detain ourselves in subtleties regarding the binary independence assumption for the model. The section on bibliographic discussion points to references which cover these details.

The probabilistic model attempts to capture the IR problem within a probabilistic framework. The fundamental idea is as follows. Given a user query, there is a set of documents which contains exactly the relevant documents and no other. Let us refer to this set of documents as the *ideal* answer set. Given the description of this ideal answer set, we would have no problems in retrieving its documents. Thus, we can think of the querying process as a process of specifying the properties of an ideal answer set (which is analogous to interpreting the IR problem as a problem of clustering). The problem is that we do not know exactly what these properties are. All we know is that there are index terms whose semantics should be used to characterize these properties. Since these properties are not known at query time, an effort has to be made at initially guessing what they could be. This initial guess allows us to generate a preliminary probabilistic description of the ideal answer set which is used to retrieve a first set of documents. An interaction with the user is then initiated with the purpose of improving the probabilistic description of the ideal answer set. Such interaction could proceed as follows.

The user takes a look at the retrieved documents and decides which ones are relevant and which ones are not (in truth, only the first top documents need to be examined). The system then uses this information to refine the description of the ideal answer set. By repeating this process many times, it is expected that such a description will evolve and become closer to the real description of the ideal answer set. Thus, one should always have in mind the need to guess at the beginning the description of the ideal answer set. Furthermore, a conscious effort is made to model this description in probabilistic terms.

The probabilistic model is based on the following fundamental assumption.

Assumption (Probabilistic Principle) Given a user query q and a document d_j in the collection, the probabilistic model tries to estimate the probability that the user will find the document d_j interesting (i.e., relevant). The model assumes that this probability of relevance depends on the query and the document representations only. Further, the model assumes that there is a subset of all documents which the user prefers as the answer set for the query q. Such an *ideal* answer set is labeled R and should maximize the overall probability of relevance to the user. Documents in the set R are predicted to be *relevant* to the query. Documents not in this set are predicted to be *non-relevant*.

This assumption is quite troublesome because it does not state explicitly how to compute the probabilities of relevance. In fact, not even the sample space which is to be used for defining such probabilities is given.

Given a query q, the probabilistic model assigns to each document d_j, as a measure of its similarity to the query, the ratio $P(d_j$ relevant-to $q)/P(d_j$ non-relevant-to $q)$ which computes the odds of the document d_j being relevant to the query q. Taking the odds of relevance as the rank minimizes the probability of an erroneous judgement [282, 785].

Definition *For the probabilistic model, the index term weight variables are all binary i.e., $w_{i,j} \in \{0,1\}$, $w_{i,q} \in \{0,1\}$. A query q is a subset of index terms. Let R be the set of documents known (or initially guessed) to be relevant. Let $\overline{R}$ be the complement of R (i.e., the set of non-relevant documents). Let $P(R|\vec{d_j})$*

be the probability that the document d_j is relevant to the query q and $P(\overline{R}|\vec{d_j})$ be the probability that d_j is non-relevant to q. The similarity $sim(d_j, q)$ of the document d_j to the query q is defined as the ratio

$$sim(d_j, q) = \frac{P(R|\vec{d_j})}{P(\overline{R}|\vec{d_j})}$$

Using Bayes' rule,

$$sim(d_j, q) = \frac{P(\vec{d_j}|R) \times P(R)}{P(\vec{d_j}|\overline{R}) \times P(\overline{R})}$$

$P(\vec{d_j}|R)$ stands for the probability of randomly selecting the document d_j from the set R of relevant documents. Further, $P(R)$ stands for the probability that a document randomly selected from the entire collection is relevant. The meanings attached to $P(\vec{d_j}|\overline{R})$ and $P(\overline{R})$ are analogous and complementary.

Since $P(R)$ and $P(\overline{R})$ are the same for all the documents in the collection, we write,

$$sim(d_j, q) \sim \frac{P(\vec{d_j}|R)}{P(\vec{d_j}|\overline{R})}$$

Assuming independence of index terms,

$$sim(d_j, q) \sim \frac{(\prod_{g_i(\vec{d_j})=1} P(k_i|R)) \times (\prod_{g_i(\vec{d_j})=0} P(\overline{k}_i|R))}{(\prod_{g_i(\vec{d_j})=1} P(k_i|\overline{R})) \times (\prod_{g_i(\vec{d_j})=0} P(\overline{k}_i|\overline{R}))}$$

$P(k_i|R)$ stands for the probability that the index term k_i is present in a document randomly selected from the set R. $P(\overline{k}_i|R)$ stands for the probability that the index term k_i is not present in a document randomly selected from the set R. The probabilities associated with the set $\overline{R}$ have meanings which are analogous to the ones just described.

Taking logarithms, recalling that $P(k_i|R) + P(\overline{k}_i|R) = 1$, and ignoring factors which are constant for all documents in the context of the same query, we can finally write

$$sim(d_j, q) \sim \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

which is a key expression for ranking computation in the probabilistic model.

Since we do not know the set R at the beginning, it is necessary to devise a method for initially computing the probabilities $P(k_i|R)$ and $P(k_i|\overline{R})$. There are many alternatives for such computation. We discuss a couple of them below.

In the very beginning (i.e., immediately after the query specification), there are no retrieved documents. Thus, one has to make simplifying assumptions such as: (a) assume that $P(k_i|R)$ is constant for all index terms k_i (typically, equal to 0.5) and (b) assume that the distribution of index terms among the non-relevant documents can be approximated by the distribution of index terms among all the documents in the collection. These two assumptions yield

$$\begin{aligned} P(k_i|R) &= 0.5 \\ P(k_i|\overline{R}) &= \frac{n_i}{N} \end{aligned}$$

where, as already defined, n_i is the number of documents which contain the index term k_i and N is the total number of documents in the collection. Given this initial guess, we can then retrieve documents which contain query terms and provide an initial probabilistic ranking for them. After that, this initial ranking is improved as follows.

Let V be a subset of the documents initially retrieved and ranked by the probabilistic model. Such a subset can be defined, for instance, as the top r ranked documents where r is a previously defined threshold. Further, let V_i be the subset of V composed of the documents in V which contain the index term k_i. For simplicity, we also use V and V_i to refer to the number of elements in these sets (it should always be clear when the used variable refers to the set or to the number of elements in it). For improving the probabilistic ranking, we need to improve our guesses for $P(k_i|R)$ and $P(k_i|\overline{R})$. This can be accomplished with the following assumptions: (a) we can approximate $P(k_i|R)$ by the distribution of the index term k_i among the documents retrieved so far, and (b) we can approximate $P(k_i|\overline{R})$ by considering that all the non-retrieved documents are not relevant. With these assumptions, we can write,

$$\begin{aligned} P(k_i|R) &= \frac{V_i}{V} \\ P(k_i|\overline{R}) &= \frac{n_i - V_i}{N - V} \end{aligned}$$

This process can then be repeated recursively. By doing so, we are able to improve on our guesses for the probabilities $P(k_i|R)$ and $P(k_i|\overline{R})$ without any assistance from a human subject (contrary to the original idea). However, we can also use assistance from the user for definition of the subset V as originally conceived.

The last formulas for $P(k_i|R)$ and $P(k_i|\overline{R})$ pose problems for small values of V and V_i which arise in practice (such as $V = 1$ and $V_i = 0$). To circumvent these problems, an adjustment factor is often added in which yields

$$\begin{aligned} P(k_i|R) &= \frac{V_i + 0.5}{V + 1} \\ P(k_i|\overline{R}) &= \frac{n_i - V_i + 0.5}{N - V + 1} \end{aligned}$$

An adjustment factor which is constant and equal to 0.5 is not always satisfactory. An alternative is to take the fraction n_i/N as the adjustment factor which yields

$$\begin{aligned} P(k_i|R) &= \frac{V_i + \frac{n_i}{N}}{V+1} \\ P(k_i|\overline{R}) &= \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1} \end{aligned}$$

This completes our discussion of the probabilistic model.

The main *advantage* of the probabilistic model, in theory, is that documents are ranked in decreasing order of their *probability* of being relevant. The *disadvantages* include: (1) the need to guess the initial separation of documents into relevant and non-relevant sets; (2) the fact that the method does *not* take into account the frequency with which an index term occurs inside a document (i.e., all weights are binary); and (3) the adoption of the independence assumption for index terms. However, as discussed for the vector model, it is not clear that independence of index terms is a bad assumption in practical situations.

2.5.5 Brief Comparison of Classic Models

In general, the Boolean model is considered to be the weakest classic method. Its main problem is the inability to recognize partial matches which frequently leads to poor performance. There is some controversy as to whether the probabilistic model outperforms the vector model. Croft performed some experiments and suggested that the probabilistic model provides a better retrieval performance. However, experiments done afterwards by Salton and Buckley refute that claim. Through several different measures, Salton and Buckley showed that the vector model is *expected* to outperform the probabilistic model with general collections. This also seems to be the dominant thought among researchers, practitioners, and the Web community, where the popularity of the vector model runs high.

2.6 Alternative Set Theoretic Models

In this section, we discuss two alternative set theoretic models, namely the fuzzy set model and the extended Boolean model.

2.6.1 Fuzzy Set Model

Representing documents and queries through sets of keywords yields descriptions which are only partially related to the real semantic contents of the respective documents and queries. As a result, the matching of a document to the query terms is approximate (or vague). This can be modeled by considering that each query term defines a *fuzzy* set and that each document has a *degree of membership* (usually smaller than 1) in this set. This interpretation of the retrieval process (in terms of concepts from fuzzy theory) is the basic foundation of the various fuzzy set models for information retrieval which have been proposed over the years. Instead of reviewing several of these models here, we focus on a particular one whose description fits well with the models already covered in this chapter. Thus, our discussion is based on the fuzzy set model for information retrieval proposed by Ogawa, Morita, and Kobayashi [616]. Before proceeding, we briefly introduce some fundamental concepts.

Fuzzy Set Theory

Fuzzy set theory [846] deals with the representation of classes whose boundaries are not well defined. The key idea is to associate a membership function with the elements of the class. This function takes values in the interval [0,1] with 0 corresponding to no membership in the class and 1 corresponding to full membership. Membership values between 0 and 1 indicate *marginal* elements of the class. Thus, membership in a fuzzy set is a notion intrinsically *gradual* instead of abrupt (as in conventional Boolean logic).

Definition *A fuzzy subset A of a universe of discourse U is characterized by a membership function $\mu_A : U \rightarrow [0,1]$ which associates with each element u of U a number $\mu_A(u)$ in the interval [0,1].*

The three most commonly used operations on fuzzy sets are: the *complement* of a fuzzy set, the *union* of two or more fuzzy sets, and the *intersection* of two or more fuzzy sets. They are defined as follows.

Definition *Let U be the universe of discourse, A and B be two fuzzy subsets of U, and $\overline{A}$ be the complement of A relative to U. Also, let u be an element of U. Then,*

$$\begin{aligned}
\mu_{\overline{A}}(u) &= 1 - \mu_A(u) \\
\mu_{A\cup B}(u) &= max(\mu_A(u), \mu_B(u)) \\
\mu_{A\cap B}(u) &= min(\mu_A(u), \mu_B(u))
\end{aligned}$$

Fuzzy sets are useful for representing vagueness and imprecision and have been applied to various domains. In what follows, we discuss their application to information retrieval.

Fuzzy Information Retrieval

As discussed in Chapters 5 and 7, one additional approach to modeling the information retrieval process is to adopt a thesaurus (which defines term relationships). The basic idea is to expand the set of index terms in the query with related terms (obtained from the thesaurus) such that additional relevant documents (i.e., besides the ones which would be normally retrieved) can be retrieved by the user query. A thesaurus can also be used to model the information retrieval problem in terms of fuzzy sets as follows.

A thesaurus can be constructed by defining a *term-term correlation matrix* $\vec{c}$ (called *keyword connection matrix* in [616]) whose rows and columns are associated to the index terms in the document collection. In this matrix $\vec{c}$, a normalized correlation factor $c_{i,l}$ between two terms k_i and k_l can be defined by

$$c_{i,l} = \frac{n_{i,l}}{n_i + n_l - n_{i,l}}$$

where n_i is the number of documents which contain the term k_i, n_l is the number of documents which contain the term k_l, and $n_{i,l}$ is the number of documents which contain both terms. Such a correlation metric is quite common and has been used extensively with clustering algorithms as detailed in Chapter 5.

We can use the term correlation matrix $\vec{c}$ to define a fuzzy set associated to each index term k_i. In this fuzzy set, a document d_j has a degree of membership $\mu_{i,j}$ computed as

$$\mu_{i,j} = 1 - \prod_{k_l \, \in \, d_j} (1 - c_{i,l})$$

which computes an algebraic sum (here implemented as the complement of a negated algebraic product) over all terms in the document d_j. A document d_j belongs to the fuzzy set associated to the term k_i if its own terms are related to k_i. Whenever there is at least one index term k_l of d_j which is strongly related to the index k_i (i.e., $c_{i,l} \sim 1$), then $\mu_{i,j} \sim 1$ and the index k_i is a good fuzzy index for the document d_j. In the case when all index terms of d_j are only loosely related to k_i, the index k_i is not a good fuzzy index for d_j (i.e., $\mu_{i,j} \sim 0$). The adoption of an algebraic sum over all terms in the document d_j (instead of the classic *max* function) allows a smooth transition for the values of the $\mu_{i,j}$ factor.

The user states his information need by providing a Boolean-like query expression. As also happens with the classic Boolean model (see the beginning of this chapter), this query is converted to its disjunctive normal form. For instance, the query $[q = k_a \wedge (k_b \vee \neg k_c)]$ can be written in disjunctive normal form as $[\vec{q}_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)]$, where each of the components is a binary weighted vector associated to the tuple (k_a, k_b, k_c). These binary weighted vectors are the conjunctive components of $\vec{q}_{dnf}$. Let cc_i be a reference to the i-th conjunctive component. Then,

$$\vec{q}_{dnf} = cc_1 \vee cc_2 \vee \ldots \vee cc_p$$

where p is the number of conjunctive components of $\vec{q}_{dnf}$. The procedure to compute the documents relevant to a query is analogous to the procedure adopted

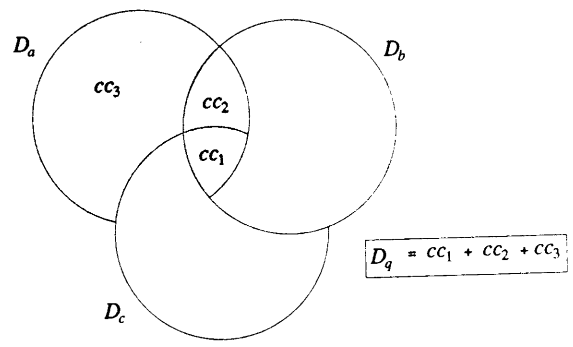

Figure 2.5 Fuzzy document sets for the query $[q = k_a \wedge (k_b \vee \neg k_c)]$. Each cc_i, $i \in \{1, 2, 3\}$, is a conjunctive component. D_q is the query fuzzy set.

by the classic Boolean model. The difference is that here we deal with fuzzy (instead of *crispy* or Boolean) sets. We proceed with an example.

Consider again the query $[q = k_a \wedge (k_b \vee \neg k_c)]$. Let D_a be the fuzzy set of documents associated to the index k_a. This set is composed, for instance, by the documents d_j which have a degree of membership $\mu_{a,j}$ greater than a predefined threshold K. Further, let $\overline{D}_a$ be the complement of the set D_a. The fuzzy set $\overline{D}_a$ is associated to $\overline{k}_a$, the negation of the index term k_a. Analogously, we can define fuzzy sets D_b and D_c associated to the index terms k_b and k_c, respectively. Figure 2.5 illustrates this example. Since the sets are all fuzzy, a document d_j might belong to the set D_a, for instance, even if the text of the document d_j does not mention the index k_a.

The query fuzzy set D_q is a union of the fuzzy sets associated with the three conjunctive components of $\vec{q}_{dnf}$ (which are referred to as cc_1, cc_2, and cc_3). The membership $\mu_{q,j}$ of a document d_j in the fuzzy answer set D_q is computed as follows.

$$\begin{aligned}
\mu_{q,j} &= \mu_{cc_1+cc_2+cc_3,j} \\
&= 1 - \prod_{i=1}^{3}(1 - \mu_{cc_i,j}) \\
&= 1 - (1 - \mu_{a,j}\mu_{b,j}\mu_{c,j}) \times \\
&\quad (1 - \mu_{a,j}\mu_{b,j}(1 - \mu_{c,j})) \times (1 - \mu_{a,j}(1 - \mu_{b,j})(1 - \mu_{c,j}))
\end{aligned}$$

where $\mu_{i,j}$, $i \in \{a, b, c\}$, is the membership of d_j in the fuzzy set associated with k_i.

As already observed, the degree of membership in a disjunctive fuzzy set is computed here using an *algebraic sum*, instead of the more common *max* function. Further, the degree of membership in a conjunctive fuzzy set is computed here using an *algebraic product*, instead of the more common *min* function. This adoption of algebraic sums and products yields degrees of membership which vary more smoothly than those computed using the *min* and *max* functions and thus seem more appropriate to an information retrieval system.

This example illustrates how this fuzzy model ranks documents relative to the user query. The model uses a term-term correlation matrix to compute correlations between a document d_j and its fuzzy index terms. Further, the model adopts algebraic sums and products (instead of *max* and *min*) to compute the overall degree of membership of a document d_j in the fuzzy set defined by the user query. Ogawa, Morita, and Kobayashi [616] also discuss how to incorporate user relevance feedback into the model but such discussion is beyond the scope of this chapter.

Fuzzy set models for information retrieval have been discussed mainly in the literature dedicated to fuzzy theory and are not popular among the information retrieval community. Further, the vast majority of the experiments with fuzzy set models has considered only small collections which make comparisons difficult to make at this time.

2.6.2 Extended Boolean Model

Boolean retrieval is simple and elegant. However, since there is no provision for term weighting, no ranking of the answer set is generated. As a result, the size of the output might be too large or too small (see Chapter 10 for details on this issue). Because of these problems, modern information retrieval systems are no longer based on the Boolean model. In fact, most of the new systems adopt at their core some form of vector retrieval. The reasons are that the vector space model is simple, fast, and yields better retrieval performance. One alternative approach though is to extend the Boolean model with the functionality of partial matching and term weighting. This strategy allows one to combine Boolean query formulations with characteristics of the vector model. In what follows, we discuss one of the various models which are based on the idea of extending the Boolean model with features of the vector model.

The *extended Boolean model*, introduced in 1983 by Salton, Fox, and Wu [703], is based on a critique of a basic assumption in Boolean logic as follows. Consider a conjunctive Boolean query given by $q = k_x \wedge k_y$. According to the Boolean model, a document which contains either the term k_x or the term k_y is as irrelevant as another document which contains neither of them. However, this binary decision criteria frequently is not in accordance with common sense. An analogous reasoning applies when one considers purely disjunctive queries.

When only two terms are considered, we can plot queries and documents in a two-dimensional map as shown in Figure 2.6. A document d_j is positioned in this space through the adoption of weights $w_{x,j}$ and $w_{y,j}$ associated with the pairs $[k_x, d_j]$ and $[k_y, d_j]$, respectively. We assume that these weights are normalized and thus lie between 0 and 1. For instance, these weights can be computed as normalized tf-idf factors as follows.

$$w_{x,j} = f_{x,j} \times \frac{idf_x}{max_i \ idf_i}$$

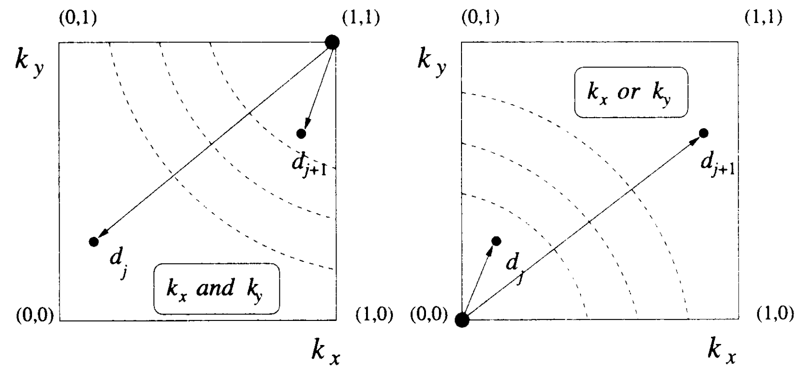

Figure 2.6 Extended Boolean logic considering the space composed of two terms k_x and k_y only.

where, as defined by equation 2.3, $f_{x,j}$ is the normalized frequency of term k_x in document d_j and idf_i is the inverse document frequency for a generic term k_i. For simplicity, in the remainder of this section, we refer to the weight $w_{x,j}$ as x, to the weight $w_{y,j}$ as y, and to the document vector $\vec{d_j} = (w_{x,j}, w_{y,j})$ as the point $d_j = (x, y)$. Observing Figure 2.6 we notice two particularities. First, for a disjunctive query $q_{or} = k_x \vee k_y$, the point $(0,0)$ is the spot to be avoided. This suggests taking the distance from $(0,0)$ as a measure of similarity with regard to the query q_{or}. Second, for a conjunctive query $q_{and} = k_x \wedge k_y$, the point $(1,1)$ is the most desirable spot. This suggests taking the complement of the distance from the point $(1,1)$ as a measure of similarity with regard to the query q_{and}. Furthermore, such distances can be normalized which yields,

$$\begin{aligned} sim(q_{or}, d) &= \sqrt{\frac{x^2 + y^2}{2}} \\ sim(q_{and}, d) &= 1 - \sqrt{\frac{(1-x)^2 + (1-y)^2}{2}} \end{aligned}$$

If the weights are all Boolean (i.e., $w_{x,j} \in \{0,1\}$), a document is always positioned in one of the four corners (i.e., (0,0), (0,1), (1,0), or (1,1)) and the values for $sim(q_{or}, d)$ are restricted to 0, $1/\sqrt{2}$, and 1. Analogously, the values for $sim(q_{and}, d)$ are restricted to 0, $1 - 1/\sqrt{2}$, and 1.

Given that the number of index terms in a document collection is t, the Boolean model discussed above can be naturally extended to consider Euclidean distances in a t-dimensional space. However, a more comprehensive generalization is to adopt the theory of vector norms as follows.

The *p-norm model* generalizes the notion of distance to include not only Euclidean distances but also p-distances, where $1 \leq p \leq \infty$ is a newly introduced parameter whose value must be specified at query time. A generalized disjunctive query is now represented by

$$q_{or} = k_1 \vee^p k_2 \vee^p \ldots \vee^p k_m$$

Analogously, a generalized conjunctive query is now represented by

$$q_{and} = k_1 \wedge^p k_2 \wedge^p \ldots \wedge^p k_m$$

The respective query-document similarities are now given by

$$\begin{aligned} sim(q_{or}, d_j) &= \left(\frac{x_1^p + x_2^p + \ldots + x_m^p}{m} \right)^{\frac{1}{p}} \\ sim(q_{and}, d_j) &= 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p + \ldots + (1-x_m)^p}{m} \right)^{\frac{1}{p}} \end{aligned}$$

where each x_i stands for the weight $w_{i,d}$ associated to the pair $[k_i, d_j]$.

The p norm as defined above enjoys a couple of interesting properties as follows. First, when $p = 1$ it can be verified that

$$sim(q_{or}, d_j) = sim(q_{and}, d_j) = \frac{x_1 + \ldots + x_m}{m}$$

Second, when $p = \infty$ it can be verified that

$$\begin{aligned} &sim(q_{or}, d_j) = max(x_i) \\ &sim(q_{and}, d_j) = min(x_i) \end{aligned}$$

Thus, for $p = 1$, conjunctive and disjunctive queries are evaluated by a sum of term-document weights as done by vector-based similarity formulas (which compute the inner product). Further, for $p = \infty$, queries are evaluated according to the formalism of fuzzy logic (which we view as a generalization of Boolean logic). By varying the parameter p between 1 and infinity, we can vary the p-norm ranking behavior from that of a vector-like ranking to that of a Boolean-like ranking. This is quite powerful and is a good argument in favor of the extended Boolean model.

The processing of more general queries is done by grouping the operators in a predefined order. For instance, consider the query $q = (k_1 \wedge^p k_2) \vee^p k_3$. The similarity $sim(q, d_j)$ between a document d_j and this query is then computed as

$$sim(q, d) = \left(\frac{\left(1 - \left(\frac{(1-x_1)^p + (1-x_2)^p}{2} \right)^{\frac{1}{p}} \right)^p + x_3^p}{2} \right)^{\frac{1}{p}}$$

This procedure can be applied recursively no matter the number of AND/OR operators.

One additional interesting aspect of this extended Boolean model is the possibility of using combinations of different values of the parameter p in a same query request. For instance, the query

$$(k_1 \vee^2 k_2) \wedge^\infty k_3$$

could be used to indicate that k_1 and k_2 should be used as in a vector system but that the presence of k_3 is required (i.e., the conjunction is interpreted as a Boolean operation). Despite the fact that it is not clear whether this additional functionality has any practical impact, the model does allow for it and does so in a natural way (without the need for clumsy extensions to handle special cases).

We should also observe that the extended Boolean model relaxes Boolean algebra interpreting Boolean operations in terms of algebraic distances. In this sense, it is really a hybrid model which includes properties of both the set theoretic models and the algebraic models. For simplicity, we opted for classifying the model as a set theoretic one.

The extended Boolean model was introduced in 1983 but has not been used extensively. However, the model does provide a neat framework and might reveal itself useful in the future.

2.7 Alternative Algebraic Models

In this section, we discuss three alternative algebraic models namely, the generalized vector space model, the latent semantic indexing model, and the neural network model.

2.7.1 Generalized Vector Space Model

As already discussed, the three classic models assume independence of index terms. For the vector model, this assumption is normally interpreted as follows.

Definition *Let $\vec{k}_i$ be a vector associated with the index term k_i. Independence of index terms in the vector model implies that the set of vectors $\{\vec{k}_1, \vec{k}_2, \ldots, \vec{k}_t\}$ is linearly independent and forms a basis for the subspace of interest. The dimension of this space is the number t of index terms in the collection.*

Frequently, independence among index terms is interpreted in a more restrictive sense to mean pairwise orthogonality among the index term vectors i.e., to mean that for each pair of index term vectors $\vec{k}_i$ and $\vec{k}_j$ we have $\vec{k}_i \bullet \vec{k}_j = 0$. In 1985, however, Wong, Ziarko, and Wong [832] proposed an interpretation in which the index term vectors are assumed linearly independent but are not pairwise orthogonal. Such interpretation leads to the *generalized vector space model* which we now discuss.

In the generalized vector space model, two index term vectors might be non-orthogonal. This means that index term vectors are not seen as the orthogonal vectors which compose the basis of the space. Instead, they are themselves composed of *smaller* components which are derived from the particular collection at hand as follows.

Definition *Given the set* $\{k_1, k_2, \ldots, k_t\}$ *of index terms in a collection, as before, let* $w_{i,j}$ *be the weight associated with the term-document pair* $[k_i, d_j]$. *If the* $w_{i,j}$ *weights are all binary then all possible patterns of term co-occurrence (inside documents) can be represented by a set of* 2^t minterms *given by* $m_1 = (0, 0, \ldots, 0)$, $m_2 = (1, 0, \ldots, 0)$, $\ldots$, $m_{2^t} = (1, 1, \ldots, 1)$. *Let* $g_i(m_j)$ *return the weight* $\{0,1\}$ *of the index term* k_i *in the minterm* m_j.

Thus, the minterm m_1 (for which $g_i(m_1) = 0$, for all i) points to the documents containing none of the index terms. The minterm m_2 (for which $g_1(m_2) = 1$, for $i = 1$, and $g_i(m_2) = 0$, for $i > 1$) points to the documents containing solely the index term k_1. Further, the minterm m_{2^t} points to the documents containing all the index terms. The central idea in the generalized vector space model is to introduce a set of pairwise orthogonal vectors $\vec{m}_i$ associated with the set of minterms and to adopt this set of vectors as the basis for the subspace of interest.

Definition *Let us define the following set of* $\vec{m}_i$ *vectors*

$$\begin{aligned}
\vec{m}_1 &= (1, 0, \ldots, 0, 0) \\
\vec{m}_2 &= (0, 1, \ldots, 0, 0) \\
&\vdots \\
\vec{m}_{2^t} &= (0, 0, \ldots, 0, 1)
\end{aligned}$$

where each vector $\vec{m}_i$ *is associated with the respective minterm* m_i.

Notice that $\vec{m}_i \bullet \vec{m}_j = 0$ for all $i \neq j$ and thus the set of $\vec{m}_i$ vectors is, by definition, *pairwise* orthogonal. This set of $\vec{m}_i$ vectors is then taken as the orthonormal basis for the generalized vector space model.

Pairwise orthogonality among the $\vec{m}_i$ vectors does not imply independence among the index terms. On the contrary, index terms are now correlated by the $\vec{m}_i$ vectors. For instance, the vector $\vec{m}_4$ is associated with the minterm $m_4 = (1, 1, \ldots, 0)$ which points to the documents in the collection containing the index terms k_1, k_2, and no others. If such documents do exist in the collection under consideration then we say that the minterm m_4 is *active* and that a dependence between the index terms k_1 and k_2 is induced. If we consider this point more carefully, we notice that the generalized vector model adopts as a basic foundation the idea that co-occurrence of index terms inside documents in the collection induces dependencies among these index terms. Since this is an idea which was introduced many years before the generalized vector space