Information Retrieval, 3, 357–377, 2000

A Hierarchical Document Retrieval Language

RONALD R. YAGER
Machine Intelligence Institute, Iona College, New Rochelle, NY 10801, USA

Received May 27, 1999; Revised March 28, 2000; Accepted May 1, 2000

Abstract. The focus of this work is on the development of a document retrieval language which attempts to enable users to better represent their requirements with respect to retrieved documents. We describe a framework for evaluating documents which allows, in the spirit of computing with words, a linguistic specification of the interrelationship between the desired attributes. This framework, which makes considerable use of the Ordered Weighted Averaging (OWA) operator, also supports a hierarchical structure which allows for an increased expressiveness of queries.

Keywords: concept formation, information retrieval, aggregation methods, hierarchical concepts

1. Introduction

The need to effectively retrieve documents satisfying user requirements has emerged as one of the most important technological problems we are now facing. While this interest is clearly driven by the explosive use of the internet, non-internet based document storage systems are also rapidly increasing. Within the current level of our capabilities, the desire for both recall and precision (Salton 1989), i.e. that we want all relevant and only relevant documents, is proving to be a very onerous burden. While future generations will almost surely have available computers with the ability to do natural language processing, a technology that will definitively solve this retrieval problem, this is not presently a viable alternative, see Grossman and Frieder (1998) for a comprehensive overview of the different approaches to information retrieval.

At the heart of the current problem with retrieval systems is the need to effectively express search requirements within a language that can be "understood" by the computer. For the most part the current paradigm involves a situation in which a document is "represented." This representation consists of a *decomposition* of a document into attributes, for each of which the document can be scored. These attributes can be as simple as the appearance of a word or phase or can be based upon a processing of the document and involve things like frequency of term occurrence or linguistic analysis. When searching, users must express their requirements in terms of these attributes. A document's conformity to the individual attributes specified by a user forms the basis of the documents overall evaluation to the user's request. Another component in the evaluation of a document is the process used to aggregate the scores of the specified attributes. The method of aggregation used reflects some aspect of the desired interrelationship between the specified attributes and as such it can be seen as a kind of *recomposition* of the document. Typical examples of aggregation are the simple average and those based upon logical connections, *Anding* and *Oring*. A considerable body of work has been devoted to the aggregation issue (Lee et al. 1992, 1993).

One way to develop more efficient retrieval systems is to provide a wide class of aggregation techniques which could enable the system to implement sophisticated interactions among attributes and thereby allow the users increased expressiveness in specifying their desires. This extension of aggregation options would be even more beneficial if, as in logic, a strong correspondence existed between formal methods of aggregation and natural language specification, a kind of computing with words (Zadeh 1996). It is our purpose here to provide such a capability.

The approach we describe here can be see to be an extension of the fuzzy set methods of information retrieval. A particularly useful resource on fuzzy methods in information retrieval is a recent survey by Kraft et al. (1999). We assume our system has a collection of basic primitive concepts. This set of primitives is similar to the term space used in the vector space model. These primitives provide a set of attributes which can be used to describe any document. If A_j is a primitive concept we can associate with any document d a value $A_j(d)$ indicating the degree to which document d is about A_j. Thus in this model a document is represented as the set of values corresponding to its score for each of the primitive concepts. In addition to providing the description of the documents the primitives are used to construct the queries to the system. For example, a user interested in documents pertaining to both A_1 and A_2 would simply pose the Boolean query (A_1 *and* A_2). The satisfaction of a document to this query is then based upon its score for these two concepts. In this work we provide tools to enable the evaluation of more sophisticated queries. For example we provide a methodology for evaluating weighted Boolean queries, $((A_1, \alpha_1)$ *and* $(A_2, \alpha_2))$, where α_1 and α_2 are the respective importances. In addition we describe a method for evaluating queries such as

I want a document that has information about <u>most</u> *of the following topics .....*

In addition the expressive capability of the query language will be enhanced by the use of a hierarchical structure to represent queries.

2. A general approach to aggregation

Central to any document retrieval system is the need for the aggregation of scores (Kraft et al. 1999). In order to provide a very general framework to implement aggregations, we shall use the Ordered Weighted Averaging (OWA) operator (Yager 1988, Yager and Kacprzyk 1997). In the following, we briefly review the basic ideas associated with this class of aggregation operators.

Definition. An Ordered Weighted Averaging (OWA) operator of dimension n is a mapping F which has an associated weighting vector

$$W = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ w_n \end{bmatrix}$$

in which

1. $w_j \in [0, 1]$
2. $\sum_{j=1}^{n} w_j = 1$

and where

$$F(a_1, a_2, \ldots, a_n) = \sum_{j=1}^{n} w_j b_i$$

with b_j being the jth largest of the a_i.

A key feature of this operator is the ordering of the arguments by value, a process that introduces a nonlinearity into the operation. Formally, we can represent this aggregation operator in vector notation as $F(a_1, a_2, \ldots, a_n) = W^T B$, where W is the weighting vector and B is a vector, called the ordered argument vector, whose components are the b_j. Here we see the nonlinearity is restricted to the process of generating B. It can be shown that this operator is in the class of mean operators (Yager 1988) as it is commutative, monotonic, and bounded, $\text{Min}[a_i] \leq F(a_1, a_2, \ldots, a_n) \leq \text{Max}[a_i]$. It can also be seen to be idempotent, $F(a, a, \ldots, a) = a$.

The great generality of the operator lies in the fact that by selecting the w_j, we can implement different aggregation operators. Specifically, by appropriately selecting the weights in W, we can emphasize different arguments based upon their position in the ordering. If we place most of the weights near the top of W, we can emphasize the higher scores, while placing the weights near bottom of W emphasizes the lower scores in the aggregation.

A number of special cases of these operators have been pointed out in the literature (Yager 1993). Each of these special cases is distinguished by the structure of the weighting vector W. Consider the situation where the weights are such that $w_1 = 1$ and $w_j = 0$ for all $j \neq 1$. This weighting vector is denoted as W^* and in this case we get $F(a_1, a_2, \ldots, a_n) = \text{Max}_j[a_j]$. Thus the Max operator is a special case of the OWA operator. If the weights are such that $w_n = 1$ and $w_j = 0$ for $j \neq n$, denoted W_*, we get $F(a_1, a_2, \ldots, a_n) = \text{Min}_j[a_j]$. Thus the Min operator is a special case of the OWA operator. As we noted above, the Min and the Max provide the extremes of this operator. If the weights are such that $w_j = \frac{1}{n}$ for all j, denoted W_{ave}, then $F(a_1, a_2, \ldots, a_n) = \frac{1}{n} \sum_{j=1}^{n} a_j$. Thus we see that the simple average is also a special case of these operators.

If $W = W^{[k]}$ is such that $w_k = 1$ and $w_j = 0$ for $j \neq k$, then $F(a_1, a_2, \ldots, a_n) = b_k$, the kth largest of the a_i. The median is also a special case of this family of operators. If n is odd, we obtain the median by selecting $w_{(n+1)/2} = 1$ and by letting $w_j = 0$, for $j \neq \frac{n+1}{2}$. If n is even, we get the median by selecting $w_{n/2} = w_{(n/2)+1} = \frac{1}{2}$ and letting $w_j = 0$ for all other terms.

An interesting class of these operators is the so-called olympic aggregators. The simplest example of this case is where we select $w_1 = w_n = 0$ and let $w_j = \frac{1}{n-2}$ for $j \neq 1$ or n. In this case, we have eliminated the highest and lowest scores and we've taken the average of the rest. We note that this process is often used in obtaining aggregated scores from judges in olympic events such as gymnastics and diving.

In Yager (1988), we introduced two measures useful for characterizing OWA operators. The first of these measures, called the alpha value of the weighting vector, is defined as

$$\alpha = \frac{1}{n-1}\sum_{j=1}^{n}(n-j)w_j.$$

It can be shown, $\alpha \in [0, 1]$. Furthermore, it can also be shown that:

$\alpha = 1$ if $W = W^*$
$\alpha = 0.5$ if $W = W_{\text{ave}}$
$\alpha = 0$ if W_*.

Essentially α provides some indication of the inclination of the OWA operators for giving more weight to the higher scores or lower scores. The closer α is to one, greater preference is given to the higher scores', the closer α to zero, the greater preference is given to lower scores, and a value close to 0.5 indicates no preference. The actual semantics associated with α depends upon the application at hand. For example, in using the OWA operators to model logical connectives between the *and* and *or*, α can be associated with a measure of the degree of *orness* associated with an aggregation. We note that if we use $W^{[k]}$, then $\alpha = \frac{n-k}{n-1}$ and we see that as k moves from one (Max) to n (Min) α gets smaller.

It can be shown that while $\alpha = 1$ only if $W = W^*$ and $\alpha = 0$ only if $W = W_*$, other values of α can be obtained for many different cases of W. A particularly interesting case is $\alpha = 0.5$. It can be shown that for any OWA operator having a W with $w_{n-j+1} = w_j$ for all j, we get $\alpha = 0.5$. Thus we see any symmetric OWA operator has $\alpha = 0.5$. Essentially these operators are in the same spirit as the simple average.

The second measure introduced in Yager (1988) was

$$\text{Disp}(W) = -\sum_{j=1}^{n} w_j \ln(w_j).$$

In Yager (1988) it was suggested that this measure can be used to measure the degree to which we use all the information in the argument. It can be shown that for all W

$$0 \leq \text{Disp}(w) \leq \ln(n).$$

We note $\text{Disp}(w) = 0$ iff $W = W_{(k)}$ and $\text{Disp}(w) = \ln(n)$ iff $W = W_{\text{ave}}$. It can be shown that of all the symmetric implementations of W, those having $\alpha = 0.5$ (W_{ave}) has the largest measure of Disp.

3. Linguistic expressions to obtain the OWA weighting vector

Let us now consider a basic application of the OWA operator in document retrieval systems. Assume $A_1, A_2, \ldots, A_n$ is a collection of attributes of interest to a searcher using a document retrieval system. For any given document d, let $A_i(d) \in [0, 1]$ indicate the degree to which

document d satisfies the property associated with attribute A_i. Using the OWA operator, we can obtain an overall valuation of document d as

$$\text{Val}(d) = F_w(A_i(d), A_2(d), \ldots, A_n(d)).$$

Since the value obtained as a result of using OWA aggregation is dependent upon the weighting vector, the issue of deciding upon the weighting vector appropriate for a particular aggregation is of great importance. A number of different approaches have been suggested for obtaining the weighting vector to use in any given application (Yager 1997). For our purpose, that of developing a user friendly document retrieval system, we shall describe an approach based upon the idea of linguistic quantifiers.

The concept of linguistic quantifiers was originally introduced by Zadeh (1983). A linguistic quantifier, more specifically a proportional linguistic quantifier, is a term corresponding to a proportion of objects. While most formal systems, such as logic, allow just two quantifiers, *for all* and *there exists*, as noted by Zadeh, human discourse is replete with a vast array of terms, fuzzy and crisp, that are used to express information about proportions. Examples of this are *most*, *at least half*, *all*, *about* 1/3. Motivated by this Zadeh (1983) suggested a method for formally representing these linguistic quantifiers. Let $\mathcal{Q}$ be a linguistic expression corresponding to a quantifier such as *most*; then, Zadeh suggested, represent this as a fuzzy subset Q over $I = [0, 1]$ in which for any proportion $r \in I$, $Q(r)$ indicates the degree to which r satisfies the concept indicated by the quantifier $\mathcal{Q}$.

In Yager (1996) Yager showed how we can use a linguistic quantifier to obtain a weighting vector W associated with an OWA aggregation. For our purposes we shall restrict ourselves to regularly increasing monotonic (RIM) quantifiers. A fuzzy subset $Q : I \to I$ is said to represent a RIM linguistic quantifier if

1) $Q(0) = 0$
2) $Q(1) = 1$
3) if $r_1 > r_2$ then $Q(r_1) \geq Q(r_2)$ (monotonic)

These RIM quantifiers model the class in which an increase in proportion results in an increase in compatibility to the linguistic expression being modeled. Examples of these types of quantifiers are *at least one*, *all*, *at least* α%, *most*, *more than a few*, *some*.

Assume Q is a RIM quantifier. Then we can associate with Q an OWA weighting vector W such that for $j = 1$ to n

$$w_j = Q\left(\frac{j}{n}\right) - Q\left(\frac{j-1}{n}\right)$$

Thus using this approach we obtain the weighting vector directly from the linguistic expression of the quantifier. The properties of RIMness guarantee that the properties of W are satisfied:

1. Since Q is monotonic, $Q(\frac{j}{n}) \geq Q(\frac{j-1}{n})$, and so $w_j \geq 0$
2. $\sum_{j=1}^{n} w_j = \sum_{j=1}^{n} Q(\frac{j}{n}) - Q(\frac{j-1}{n}) = Q(1) - Q(0) = 1$

Figure 1. Linguistic quantifier "for all".

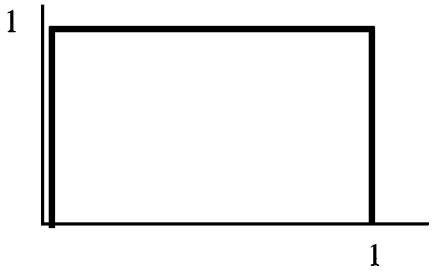

Figure 2. Linguistic quantifier "not none".

Let us look at the situation for some prototypical quantifiers. The quantifier *for all* is shown in figure 1. In this case we get that $w_j = 0$ for $j \neq n$, and $w_n = 1$, $W = W_*$. In this case we get as our aggregation the minimum of the aggregates. We also recall that the quantifier *for all* corresponds to the logical "anding" of all the arguments.

In figure 2 we see the existential quantifier, *at least one*, this the same as *not none*. In this case $w_1 = 1$ and $w_j = 0$ for $j > 1$, $W = W^*$. This can be seen as inducing the maximum aggregation. It is recalled this quantifier corresponds to a logical *oring* of the arguments.

Figure 3 is seen as corresponding to the quantifier *at least* α. For this quantifier $w_j = 1$ for j such that $\frac{j-1}{n} < \alpha \leq \frac{j}{n}$ and $w_j = 0$ for all other.

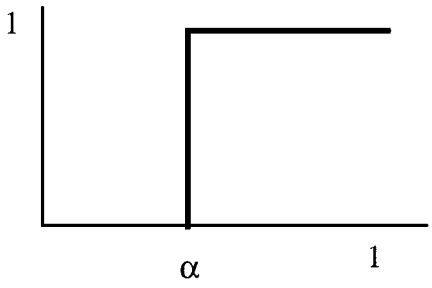

Figure 3. Linguistic quantifier "at least α".

Another quantifier is one in which $Q(r) = r$ for $r \in [0, 1]$. For this quantifier we get $w_j = \frac{j}{n} - \frac{j-1}{n} = \frac{1}{n}$ for all j. This gives us the simple average. We shall denote this quantifier as *some*.

As discussed by Yager (1993) one can consider parameterized families of quantifiers. For example consider the parameterized family $Q(r) = r^\rho$, where $\rho \in [1, \infty]$. Here if $\rho = 0$, we get the existential quantifier; when $\rho = \infty$, we get the quantifier *for all* and when $\rho = 1$, we are get the quantifier *some*. In addition for the case in which $\rho = 2$, $Q(r) = r^2$, we get one possible interpretation of the quantifier *most*.

We are now in a position to address the issue of obtaining the OWA weighting vector to be used in a search in a user friendly document retrieval system. In constructing such a user friendly system we shall make available to the user a vocabulary, $\mathcal{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_q\}$, of linguist expressions, each corresponding to a linguistic quantifier. When posing a query, the user, after specifying a collection of attributes of interest $(A_1, A_2, \ldots, A_n)$, will be prompted to also specify one of the linguistic quantifiers in $\mathcal{Q}$ as guiding the query formation. Transparent to the user is the association of each of the linguistic terms in $\mathcal{Q}$, with a representative fuzzy subset, $\mathcal{Q}_i \Leftrightarrow Q_i$, and the process of converting this fuzzy subset into an OWA weighting vector based on the formulation

$$w_j = Q_i\left(\frac{j}{n}\right) - Q_i\left(\frac{j-1}{n}\right).$$

One of the elements in $\mathcal{Q}$ should be designated as the default quantifier, this is the one that is to be used when no selection is specified by the user. Perhaps the most appropriate choice for this is the average quantifier $w_j = \frac{1}{n}$, which corresponds to the linguistic expression *some*.

The process of actually selecting the set $\mathcal{Q}$ is beyond our scope here and clearly will benefit from some empirical research.

As a result of the ideas so far presented here we can introduce the idea of a query module: $\langle A_1, A_2, \ldots, A_n : \mathcal{Q}\rangle$, consisting of a collection of attributes required of the document and a linguistic quantifier indicating the proportion of the attributes we desire. Implicit in this querymodule is the fact the the linguistic expression $\mathcal{Q}$ is essentially defining a weighting vector W for an OWA aggregation.

4. Including attribute importance in queries

In the preceding we have indicated a query object to be a module consisting of a collection of attributes of interest and a quantifier Q indicating a mode of interaction between the attributes. As noted the quantifier is to be selected from a collection of quantifiers, among which are the logical "anding" of the attribute scores, the logical "oring" of the attribute scores, and the simple averaging of attribute scores. Due to this generality we can accommodate in our framework both logical type of retrieval systems (Negoita and Flonder 1976, Kraft and Buell 1983, Meadow 1992) and as well the vector space type described by Salton (1989).

Implicit in the preceding is the equal treatment of all desired attributes. Often a user may desire to ascribe different weights or importances to the different attributes (Kraft and Buell

1983, Yager 1987, Dubois et al. 1988, Sanchez 1989). In the following we shall consider the introduction of importance weights into our procedure.

Let $\alpha_i \in [0, 1]$ be a value associated with an attribute indicating the importance associated with the attribute. We shall assume the larger α_i the more important attribute i is to the user and let $\alpha_i = 0$ stipulate zero importance. With the introduction of these weights we can now consider a more general query object:

$$\langle A_1, A_2, \ldots, A_n : M : Q \rangle.$$

Here as before, the A_i are a collection of attributes and Q is a linguistic quantifier, however, here M is an n vector whose component $m_j = \alpha_j$, the importance associated with A_j.

Our goal now is to calculate the overall score $\mathrm{Val}(d)$ associated with a document d, we shall denote this

$$\mathrm{Val}(d) = F_{Q/M}(A_1(d), A_2(d), \ldots, A_n(d))$$

Here $F_{Q/M}$ indicates an OWA operator. Our agenda here will be to first find an associated OWA weighting vector, $W(d)$, based upon both Q and M. Once having obtained this vector we calculate $\mathrm{Val}(d)$ by the usual OWA process

$$\mathrm{Val}(d) = W(d)^T B(d) = \sum_{j=1}^{n} w_j(d) b_j(d)$$

Here $b_j(d)$ is denoting the jth largest of the $A_i(d)$ and $w_j(d)$ is the jth component of the associated OWA vector $W(d)$.

What is important to point out here is that, as we shall subsequently see, as opposed to the original case, where no importances are considered, the associated OWA vector will be different for each d. This situation accounts for our denotation $W(d)$. Actually the weighting vector will be influenced by the ordering of the $A_i(d)$.

We now describe the procedure (Yager 1996, 1997) that shall be used to calculate the weighting vector, $w_j(d)$. The first step is to calculate the ordered argument vector $B(d)$ such that $b_j(d)$ is the jth largest of the $A_j(d)$. Furthermore, we shall let μ_j denote the importance weight associated with the attribute that has the jth largest value. Thus if $A_5(d)$ is the largest of the $A_i(d)$, then $b_1(d) = A_5(d)$ and $u_1 = \alpha_5$. Our next step is to calculate the OWA weighting vector $W(d)$. We obtain the associated weights as

$$w_j(d) = Q\left(\frac{S_j}{T}\right) - Q\left(\frac{S_j - 1}{T}\right)$$

where $S_j = \sum_{k=1}^{j} u_k$ and $T = S_n = \sum_{k=1}^{n} u_k$. Thus T is the sum of all the importances and S_j is the sum of the importances of the jth most satisfied attributes. Once having obtained these weights we can then obtain the aggregated value by the usual method, $B^T W$. The following example will illustrate the use of this technique.

Example. We shall assume there are four criteria of interest to the user: A_1, A_2, A_3, A_4. The importances associated with these criteria are $X_1 = 1$, $X_2 = 0.6$, $X_3 = 0.5$ and $X_4 = 0.9$.

From this we get $T = \sum_{k=1}^{4} X_k = 3$. We shall assume the quantifier guiding this aggregation is *most*, which is defined by $Q(r) = r^2$. Assume we have two documents x and y and the satisfactions to each of the attributes by the documents is given by the following:

$$A_1(x) = 0.7 \quad A_2(x) = 1 \quad A_3(x) = 0.5 \quad A_4(x) = 0.6$$
$$A_1(y) = 0.6 \quad A_2(y) = 0.3 \quad A_3(y) = 0.9 \quad A_4(y) = 1$$

Our objective here is to obtain the valuations of each of the documents with respect to this query structure. We first consider the valuation for x. In this case the ordering of the criteria satisfactions gives us:

	b_j	u_j
A_2	1	0.6
A_1	0.7	1
A_4	0.6	0.9
A_3	0.5	0.5

Calculating the weights associated with x, which we denoted $w_i(x)$, we get

$$w_1(x) = Q\left(\frac{0.6}{3}\right) - Q\left(\frac{0}{3}\right) = 0.04$$
$$w_2(x) = Q\left(\frac{1.6}{3}\right) - Q\left(\frac{0.6}{3}\right) = 0.24$$
$$w_3(x) = Q\left(\frac{2.5}{3}\right) - Q\left(\frac{1.6}{3}\right) = 0.41$$
$$w_4(x) = Q\left(\frac{3}{3}\right) - Q\left(\frac{1.6}{3}\right) = 0.31$$

Using this we calculate $\mathrm{Val}(x)$

$$\mathrm{Val}(x) = \sum_{j=1}^{4} w_j(x)\, b_j = (.04)(1) + (.24)(.7) + (.41)(.6) + (.31)(.5) = 0.609$$

To calculate the score for document y we proceed as follows. In this case the ordering of the criteria satisfaction is

	b_j	u_j
A_4	1	0.9
A_3	0.9	0.5
A_1	0.6	1
A_2	0.3	0.6

The weights associated with the aggregation are

$$w_1(y) = Q\left(\frac{0.9}{3}\right) - Q\left(\frac{0}{3}\right) = 0.09$$

$$w_2(y) = Q\left(\frac{1.4}{3}\right) - Q\left(\frac{0.9}{3}\right) = 0.13$$

$$w_3(y) = Q\left(\frac{2.4}{3}\right) - Q\left(\frac{1.4}{3}\right) = 0.42$$

$$w_4(y) = Q\left(\frac{3}{3}\right) - Q\left(\frac{2.4}{3}\right) = 0.36$$

Using this we calculate

$$\text{Val}(y) = \sum_{j=1}^{4} w_j(y)\, b_j = (.09)(1) + (.13)(.9) + (.42)(.6) + (.36)(.3) = 0.567$$

Hence in this example x is the better scoring document.

It is important to observe that the weights are different for the two aggregations. This is due to the fact that the ordering of the satisfactions to the A_i's are different for x and y which leads to a different ordering of the u_j's resulting in a different weighting vector.

More details with respect to the properties of this methodology can be found in Yager (1996, 1997), however here we shall point out some properties associated with this approach.

Any attribute that has importance weight zero has no affect on the result. Without loss of generality, we shall assume the indexing of the A_i have been such that $A_i(d) \geq A_j(d)$ if $i < j$. In this case $w_j(d) = Q(\frac{1}{T}\sum_{k=1}^{j}\alpha_k) - Q(\frac{1}{T}\sum_{k=1}^{j-1}\alpha_k)$. If $\alpha_j = 0$ then $w_j = 0$ and $w_i A_i(d) = 0$ no matter what value $A_i(d)$. It can be easily seen that if we eliminate A_j, we get the same result as that obtained with $\alpha_j = 0$.

Consider the situation when all the attributes have the same importance, $\alpha_j = \alpha$. In this case

$$w_j(d) = Q\left(\frac{1}{n\alpha}\sum_{k=1}^{j}\alpha\right) - Q\left(\frac{1}{n\alpha}\sum_{k=1}^{j-1}\alpha\right) = Q\left(\frac{j}{n}\right) - Q\left(\frac{j-1}{n}\right).$$

This is the same set of weights we obtained when we didn't include any information with respect to importance.

We shall call a quantifier a binary quantifier if there exists some $r^* \in [0, 1]$ such that

$$Q(r) = 0 \quad \text{for } r < r^*$$
$$Q(r) = 1 \quad \text{for } r \geq r^*$$

We note the universal and existential quantifiers are binary. Consider the weights obtained from this type of quantifier, $w_j(d) = Q(\frac{1}{T}\sum_{k=1}^{j}\alpha_k) - Q(\frac{1}{T}\sum_{k=1}^{j-1}\alpha_k)$. What is clear is

that the weights will always be binary—that is there will exist some value j^* for which $w_j^* = 1$ and $w_j = 0$ for all $j \neq j^*$. While the value of j^* will depend upon the objects being aggregated it will still only have all weights equal zero except one. Because of this the aggregated value will always be equal to one of the attribute values. It can be shown that these quantifiers always have weights in which the dispersion is zero.

Let us now look at the form of aggregation function obtained for some special cases of linguistic quantifiers. In the following we shall assume, without loss of generality, that the indexing is such that $A_i(d) \geq A_j(d)$ if $i < j$. Furthermore we shall suppress the d and denote $A_i(d) = a_i$. Using this notational convention

$$\text{Val}(d) = F_{Q/\alpha}(a_1, a_2, \ldots, a_n) = \sum_{j=1}^{n} a_j w_j$$

where $w_j = Q(\frac{1}{T}\sum_{k=1}^{j} \alpha_k) - Q(\frac{1}{T}\sum_{k=1}^{j-1} \alpha_k)$.

Consider first the case of the quantifier *some*, $Q(r) = r$. For this quantifier $w_j = \alpha_j / T$ and hence

$$\text{Val}(d) = \frac{1}{T}\sum_{j=1}^{n} \alpha_j a_j$$

This is simply the weighted average of the attributes.

Consider now the case of the quantifier *for all*, $Q(1) = 1$ and $Q(r) = 0$ for $r \neq 1$. In this case $w_j = 0$ unless $\sum_{k=1}^{j} \alpha_k = T$ and $\sum_{k=1}^{j} \alpha_k < T$. From this we see $w_j = 1$ for the attribute having the smallest satisfaction and non-zero importance. Thus here,

$$\text{Val}(d) = \underset{\alpha_j \neq 0}{\text{Min}} [a_j]$$

For the case of the *existential* quantifier, $Q(0) = 0$ and $Q(r) = 1$ for all $r \neq 0$, we can easily show that

$$\text{Val}(d) = \underset{\alpha_j \neq 0}{\text{Max}} [a_j]$$

These two quantifiers are, of course, examples of what we call binary quantifiers, the first being one in which $r^* = 1$ and the second, $r^* = 0$.

Another interesting example of a binary quantifier is the median quantifier. Here $Q(r) = 0$ for $r < 0.5$ and $Q(r) = 1$ for $r \geq 0.5$. In this case it can be shown that $\text{Val}(d)$ can be obtained by the following simple process. First we normalize the weights, $\hat{\alpha}_j = \frac{\alpha_j}{T}$. Next we order the attribute scores in descending order and associate with each its normalized weight. We then, starting from the top, the highest score, add the normalized weights until we first reach a total of 0.5, the score of that attribute at which this total is reached is the aggregated value. The following example illustrates this procedure.

Example. In our preceding example, we have for x

	b	α	$\hat{\alpha}$	Sum $\hat{\alpha}$	
A_2	1	0.6	0.2	0.2	
A_1	0.7	1	0.33	0.55	←
A_4	0.6	0.9	0.3		
A_3	0.5	0.5	0.167		

Since the total goes over 0.5 when $b_j = 0.7$, we get $\text{Val}(x) = 0.7$. For y in our preceding example:

	b	α	$\hat{\alpha}$	Sum $\hat{\alpha}$	
A_4	1	0.9	0.3	0.3	
A_3	0.9	0.5	0.167	0.467	
A_1	0.6	1	0.333	0.8	←
A_2	0.3	0.6	0.2		

Here the total goes over 0.5 when $b_j = 0.6$.

An interesting example of an OWA aggregation is the so called olympic aggregation. Here $w_1 = w_n = 0$ and $w_j = \frac{1}{n-2}$ for $j \neq 1$ or n. Using this aggregation we eliminate the highest and lowest scores and then take the average of the remaining scores. We can provide a generalization of this type of aggregation using a quantifier shown in figure 4. For this case

$$
\begin{aligned}
&Q(r) = 0 && r < \rho \\
&Q(r) = \frac{r - \rho}{1 - 2\rho} && \rho \leq r \leq 1 - \rho \\
&Q(r) = 1 && r > 1 - \rho
\end{aligned}
$$

For this quantifier $w_j = 0$ for all j for which $\sum_{k=1}^{j} \frac{\alpha_k}{T} < \rho$. Similarly, $w_j = 0$ for all j for which $\sum_{k=j}^{n} \frac{\alpha_k}{T} < \rho$. In the range in between $w_j = \frac{\alpha_j}{1-2\rho}$.

Another interesting example of OWA aggregation, one that is in some sense a dual of the olympic aggregation, is the so called Arrow-Hurwicz aggregation (Arrow and Hurwicz

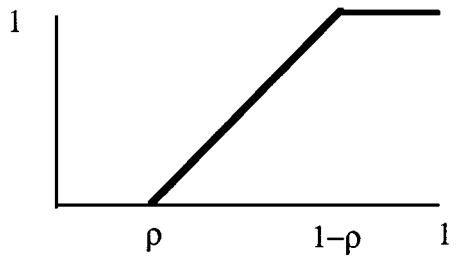

Figure 4. Generalized olympic quantifier.

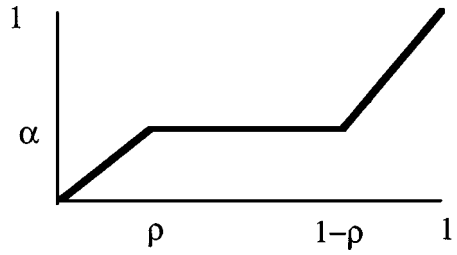

Figure 5. Generalized Arrow-Hurwicz.

1972). Here $w_1 = \alpha$ and $w_n = 1-\alpha$, and $w_j = 0$ for all other. In this case we just consider the extreme values and eliminate the middle values. We can provide a generalization of this type of aggregation, one that can be used with importance weighted attributes, by introducing the quantifier shown in figure 5. For this quantifier

$$\begin{aligned} Q(r) &= \frac{\alpha}{\rho} r && r < \rho \\ Q(r) &= \alpha && \rho \leq r < 1-\rho \\ Q(r) &= 1 - \frac{1-\alpha}{\rho}(1-r) && r \geq 1-\rho \end{aligned}$$

it is assumed $\rho \leq 0.5$. For this quantifier the weights used in the OWA aggregation are such that for the highest scoring attributes, those accounting for ρ portion of the importance, $w_j = \frac{\alpha}{\rho}$, for the least satisfied attributes, those accounting for ρ portion of the importance, $w_j = \frac{1-\alpha}{\rho}$ and the middle scoring attributes $w_j = 0$. In this quantifier α can be seen as a degree of optimism and $1-\rho$ as an indication of the extremism of the aggregation. A number special cases of this quantifier are worth noting. If $\rho = 0$ then we have $w_1 = \alpha$ and $w_n = 1-\alpha$, the basic Arrow-Hurwicz aggregation. If $\alpha = \rho = 0.5$ then we get the quantifier $Q(r) = r$. If $\alpha = 1$ then we get the quantifier *at least* ρ and if $\alpha = 0$ then we get the quantifier *at least* $1-\rho$.

5. Including priorities between attributes

In the preceding we have described a method for evaluating the overall score of documents based upon a query object:

$$\langle A_1, A_2, \ldots, A_n : M : Q \rangle$$

In this object the component α_j of the vector M indicates the weight associated with the attribute A_j. Implicit in our formulation was the idea that the weight α_j was explicitly provided by the user. This is not necessarily required. It is possible for the weight associated with attribute A_j to be determined by some property of the document itself. Thus let B_j

be some measurable attribute associated with the document, and let $B_j(d)$ be the degree to which document d satisfies this attribute. Then without any additional complexity we can allow $\alpha_j(d) = B_j(d)$. Thus here the weight associated with attribute A_j depends upon the document itself via the value $B_j(d)$. Thus within this framework we have the option of specifying the importance weights conditionally or non-conditionally or not at all. It should be noted we could of course let $\alpha_j(d) = \alpha_j B_j(d)$, that is some proportion of $B_j(d)$.

Typically the association of importance weights with attributes indicates some measure of trade-off between the worth of the attributes. For example, consider the averaging quantifier where $\text{Val}(d) = \sum_{j=1}^{n} A_j \alpha_j$. Here we see that a gain of Δ in A_j results in an increase in overall evaluation of $\alpha_j \Delta$, while a gain of Δ in A_i is worth an increase of $\alpha_i \Delta$. This of course manifests itself in the ordering of the documents. In particular, if $\alpha_j = 2$ and $\alpha_i = 1$, then we are willing to trade a gain of Δ in A_j for a loss of less than 2Δ in A_i. In some cases where we desire two attributes, we may not be willing to trade-off of for the other. For example, in designing a car, while we would like both safety and low-cost, we are not willing to give up safety for low cost. Such a situation is characterized as one in which a *priority* exists between the attributes, safety has priority over cost.

In document retrieval systems, it may be possible that we might also desire to include a priority relationship between attributes. In the following we shall suggest a mechanism that allows for the inclusion of a priority type effects.

Assume A_1 and A_2 are two attributes for which there exists a priority relationship: A_1 has priority over A_2. In order to manifest this relationship, we allow the importance associated with A_2 to be dependent upon the satisfaction of attribute A_1. Here then $\alpha_2(d) = A_1(d)$. Let us investigate this first for the simple weighted average. Assuming α_1 is fixed, we get

$$\text{Val}(d) = \alpha_1 A_1(d) + A_1(d) A_2(d) = A_1(d)(\alpha_1 + A_2(d))$$

Here we see that if $A_1(d)$ is low, the contribution of $A_2(d)$ becomes small and hence it is not possible for a high value of A_2 to compensate.

More generally, consider the quantifier Q and assume A_i has priority over A_j. To implement this priority we make the importance associated with A_j related to the satisfaction of A_i. In particular, we let $\alpha_j = \alpha A_i$, where $\alpha \in [0, 1]$. Using this we get for the weight w_j associated with A_j that

$$w_j = Q\left(S_{k-1} + \frac{\alpha A_i(d)}{T}\right) - Q(S_{k-1})$$

where $S_{k-1} = \frac{\sum_{k=1}^{j-1} \alpha_k}{T}$. We see that as $A_i(d)$ gets smaller, the value w_j will decrease.

6. Concepts and hierarchies

In the preceding we have considered the problem of document retrieval within the following framework. We have assumed a set of documents D, called the document base, from which we are interested in retrieving. We have associated with this document base a collection of attributes A_i, $i = 1$ to n. These attributes are characterized by the fact that for any $d \in D$,

we have available $A_i(d) \in [0, 1]$. More specifically, we assume no calculation is necessary to obtain A_i, we shall say that the value of attribute A_j is directly accessible. We shall now associate with a document base a slightly more general idea which we shall call a *concept*. We define a concept associated with D as an object whose measure of satisfaction, as a number in the unit interval, can be obtained for any document in D. It is clear that the attributes are examples of concepts, they are special concepts in that their values are directly accessible from the document base.

Consider now a query object of the type we have previously introduced. This is an object of the form $\langle A_1, A_2, \ldots, A_q : M : Q\rangle$. As we have indicated, the measure of satisfaction of this object for any $d \in D$ can be obtained by our aggregation process. In the light of this observation, we can consider this query object to be a concept, with

$$\text{Con} = \langle A_1, A_2, \ldots, A_q : M : Q\rangle$$

then

$$\text{Con}(d) = F_{Q/M}(A_1(d), A_2(d), \ldots, A_q(d)).$$

Thus a query object is a concept. A special concept is an individual attribute,

$$\text{Con} = \langle A_j : M : Q\rangle = A_j,$$

we shall call these atomic concepts. These atomic concepts require no Q or M, but just need an A_j specification.

Let us look at the query object type concept in more detail. The basic components in these objects are the attributes, the A_j. However, from a formal point of view, the ability to evaluate the query objects-concept is based upon the fact that for each A_j, we have a value for any d, $A_j(d)$. As we have just indicated, a concept also has this property, for any d we can obtain a measure of its satisfaction. This observation allows us to extend our idea of query object-concept to allow for concepts whose evaluation depends upon other concepts. Thus we can consider concepts of the form

$$\text{Con} = \langle \text{Con}_1, \text{Con}_2, \ldots, \text{Con}_n : M : Q\rangle.$$

Here each of the Con_j are concepts used to determine the satisfaction of Con by an aggregation process where M determines the weight of each of the participating concepts and Q is the quantifier guiding the aggregation of the component concepts.

The introduction of concepts into the query objects results in a hierarchical structure for query formation. Essentially, we unfold until we end up with queries made up of just attributes which we can directly evaluate. The following simple examples illustrate the structure.

Example. Consider here the query

(A_1 and A_2 and A_3) or (A_3 and A_4).

We can consider this as a concept

$$\langle \mathrm{Con}_1, \mathrm{Con}_2 : M : Q \rangle.$$

Here Q is the existential quantifier and $M = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. In addition

$$\mathrm{Con}_1 = \langle A_1, A_2, A_3 : M_1 : Q_1 \rangle$$
$$\mathrm{Con}_2 = \langle A_3, A_4 : M_2 : Q_2 \rangle$$

Here $Q_1 = Q_2 = \mathit{all}$ and

$$M_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad M_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

This query can be expressed in a hierarchical fashion as shown in figure 6.

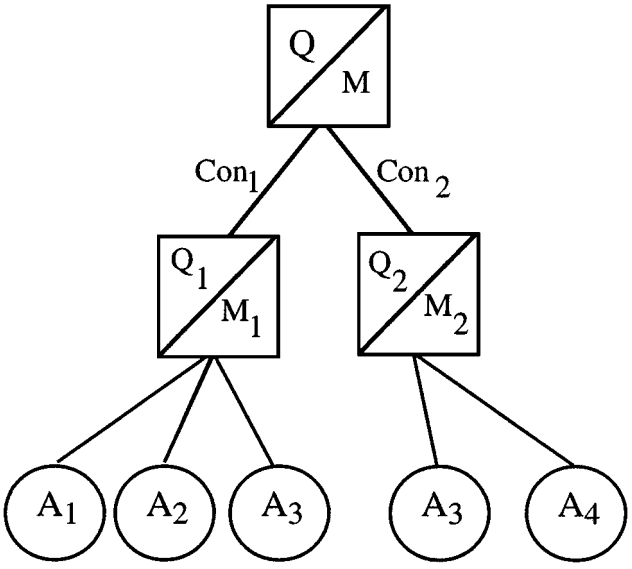

Figure 6. Hierarchical formulation of query.

7. Hierarchical querying in information retrieval

Using these ideas, we describe a hierarchical querying framework that can be used for document retrieval, we shall call this the Hierarchical Document Retrieval Language and

use the acronym HI-RET. This language can be used to retrieve documents from the Internet, or an intranet, or any other computer based environment.

Associated with any implementation of this language is a set $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ of atomic attributes, words or concepts. These atomic concepts are such that for any document d in D and any concept A_j in $\mathcal{A}$ we have directly available the value $A_j(d) \in [0, 1]$, the satisfaction of attribute A_j by document d. This information can be stored in a database such that each record is a tuple consisting of the values $A_j(d)$ for $j = 1$ to n and the address of document d. Essentially each document can be viewed as a n vector whose components are the $A_j(d)$.

In addition to the attributes, we also assume associated with any implementation of HI-RET a vocabulary of linguistic quantifiers, $\mathcal{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_q\}$ available to the searcher. Within this set of quantifiers we should surely have the quantifiers *all*, *any* and *some*. One quantifier should be designated as the default quantifier. Perhaps a best choice for this is the quantifier *some*. Transparent to the user is a fuzzy subset Q_i on the unit interval associated with each linguistic quantifier $\mathcal{Q}_i$. This fuzzy subset is used to generate the associated weights used in the aggregation.

A query to the document retrieval system is indicated by the specification of a "concept" that the user desires satisfied. The user is asked to "define" this concept by expressing it in terms of a query object, $\langle C_1, C_2, \ldots, C_n : M : \mathcal{Q}\rangle$, consisting of a group of components C_j, an importance weight associated with each of the components, M, and a quantifier, $\mathcal{Q}$ expressing the imperative for aggregating the components. The specification of the importance weights as well as quantifier are optional. If weights are not expressed, then by default they are assumed to have an importance value of one. If the quantifier is not expressed, then the designated default quantifier is assumed.

For each of the components of the query that are not an atomic object the searcher is asked to provide a definition. This process is continued until the complete hierarchy defining the query is formulated. It is noted that this hierarchy is a tree like structure in which the leaves are atomic components. Figure 7 shows a prototypical example of such a query.

Once having obtained the HI-RET expansion of a query as in figure 7, we can then use our aggregation methods to evaluate the query for each document. For example, in the case of query described in figure 7 for document d we have

$$\begin{aligned}
\mathrm{Con}_4(d) &= F_{M4/Q4}(A_6(d), A_3(d)) \\
\mathrm{Con}_3(d) &= F_{M3/Q3}(A_2(d), A_5(d), A_9(d)) \\
\mathrm{Con}_2(d) &= F_{M2/Q2}(\mathrm{Con}_4(d), A_8(d)) \\
\mathrm{Con}_1(d) &= F_{M1/Q1}(A_7(d), \mathrm{Con}_2(d), \mathrm{Con}_3(d))
\end{aligned}$$

Depending upon the skill of the user, we can capture very sophisticated queries within this framework.

An often used construct in query formulation is the logical *if* ... *then* specification expressing the desire for some attribute if some other attribute is present. In a the following we describe a method for modeling this type of structure within our HI-RET language.

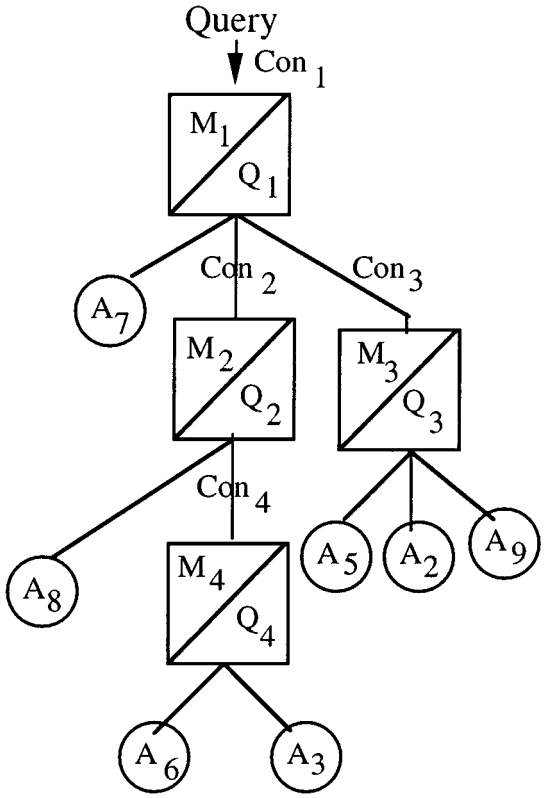

Figure 7. Prototypical query in HI-RET.

Consider the query

$(A_1$ and $A_2)$ or (if A_3 then A_4).

Figure 8 provides the hierarchical expansion of this query within the framework of HI-RET

In constructing this hierarchical implementation, we used the fact that if A_3 then A_4 is logically equivalent to not (A_3) or A_4. Thus in this proposed framework we shall interpret the concept "if A then B" as the concept $\bar{A}$ or B. We note that $\bar{A}(d) = 1 - A(d)$. More generally, the expression

if A_1 and A_2 and A_3 then B

is seen as equivalent to the expression $\bar{A}_1$ *or* $\bar{A}_2$ *or* $\bar{A}_3$ *or* B. This is represented as a query object of the form $\langle \bar{A}_1, \bar{A}_2, \bar{A}_3, B : \text{-:}\mathcal{O}r \rangle$. We note the importances have not been specified and hence by default are all assumed to be one.

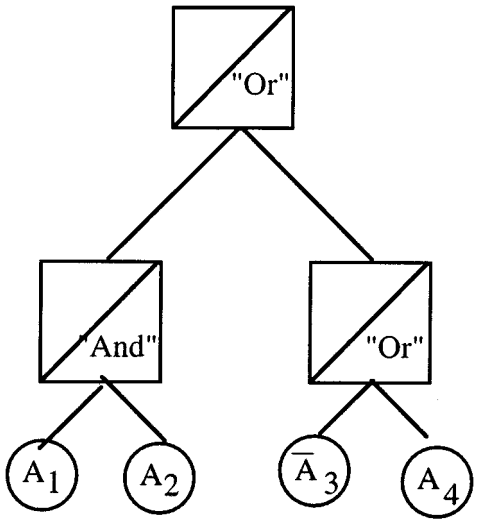

Figure 8. Implementation of query with if . . . then.

8. Thesaurus

Many document retrieval systems have a thesaurus over the set of atomic attributes (or vocabulary), a thesaurus being information about similarity between words (Larsen and Yager 1993). The function of a thesaurus is to allow for the consideration of satisfaction to synonyms when trying to evaluate the satisfaction of an attribute by a document. This function of a thesaurus can be represented within the framework presented. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be a collection of concepts corresponding to our atomic or primary attributes (vocabulary). A thesaurus can be represented as a relationship T on $\mathcal{A} \times \mathcal{A}$ such that for each pair $A_i, A_j \in \mathcal{A}$, $T(A_i, A_j) \in [0, 1]$ indicates the degree of similarity that A_j has to A_i. The basic properties of a thesaurus are:

1. Identity: $T(A_i, A_i) = 1$ for all A_i,
2. Symmetry: $T(A_i, A_j) = T(A_j, A_i)$.

If A_j is an atomic attribute we shall let $\hat{A}_j$ indicate the concept corresponding to an extended definition of the original attribute A_j. We define $\hat{A}_j$ as follows

$$\hat{A}_j = \langle \mathrm{EX}[A_j \mid A_1], \mathrm{EX}[A_j \mid A_2], \ldots, \mathrm{EX}[A_j \mid A_n] : \text{-: “Or”} \rangle,$$

where $\mathrm{EX}[A_j \mid A_i]$, the extension of A_j by A_i, is a concept defined as

$$\mathrm{EX}[A_j \mid A_i] = \langle T(A_j, A_i), A_i : \text{-:“And”} \rangle.$$

We see that the concept $\mathrm{EX}[A_j \mid A_i]$ has two components, A_i and $T(A_j, A_i)$, it uses equal importances and uses the quantifier *and*. The component $T(A_j, A_i)$ is directly accessible from the definition of the thesaurus and as such can be viewed as an atomic component. We further note that $\hat{A}_j$, a concept whose components are the n concepts $\mathrm{EX}[A_j \mid A_i]$, uses the default importance and uses the quantifier *or*.

Let us look at the form of $\hat{A}_j(d)$ resulting from this structure. First we see that

$$\mathrm{EX}[A_j \mid A_i](d) = F_Q[T(A_j, A_i), A_i(d)]$$

where $Q =$ "and." The importances are assumed to be equal. This gives that

$$\mathrm{EX}[A_j \mid A_i](d) = \mathrm{Min}[T(A_j, A_i), A_i(d)] = T(A_j, A_i) \wedge A_i(d)$$

Next we see that

$$\hat{A}_j(d) = F_{\text{“any”}}(\mathrm{EX}[A_j \mid A_1](d), \mathrm{EX}[A_j \mid A_2](d), \ldots, \mathrm{EX}[A_j \mid A_n](d))$$

this gives us

$$\hat{A}_j(d) = \operatorname*{Max}_{i=1}^{n}(\mathrm{EX}[A_j \mid A_i](d))$$

$$\hat{A}_j(d) = \operatorname*{Max}_{i=1}^{n}(T(A_j, A_i) \wedge A_i(d)]).$$

We note since $T(A_j, A_i) = 1$, then

$$\hat{A}_j(d) = A_j(d) \vee \operatorname*{Max}_{i \neq j} [T(A_j, A_i) \wedge A_i(d)]$$

When using the HI-RET language to specify a query the user can indicate whether they want to use the thesaurus or not.

9. Conclusion

The focus of this work has been on the development of a document retrieval language which enables the user to better represent their requirements with respect to retrieved documents by using appropriate aggregation operators. We have described a method, based on the OWA operators, for evaluating documents which allowed a linguistic specification of the interrelationship between the desired attributes. This framework developed here has also been shown to support aggregation in a hierarchical structure, which allows for an increased expressiveness of queries.

References

Arrow KJ and Hurwicz L (1972) An optimality criterion for decision making under ignorance. In: Carter CF and Ford JL, Eds., Uncertainty and Expectations in Economics. Kelley, New Jersey.

Dubois D, Prade H and Testemale C (1988) Weighted fuzzy pattern matching. Fuzzy Sets and Systems, 28:313–331.
Grossman DA and Frieder O (1998) Information Retrieval. Kluwer Academic Publishers, Boston.
Kraft DH, Bordogna G and Pasi G (1999) Fuzzy set techniques in information retrieval. In: Bezdek JC, Dubois D and Prade H, Eds., Fuzzy Sets in Approximate Reasoning and Information Systems. Kluwer Academic Publishers, Norwell, MA, pp. 469–510.
Kraft DH and Buell DA (1983) Fuzzy sets and generalized Boolean retrieval systems. International Journal of Man-Machine Studies, 19:45–56.
Larsen HL and Yager RR (1993) The use of fuzzy relational thesauri for classifactory problem solving in information retrieval and expert systems. IEEE Transactions on Systems, Man and Cybernetics, 23:31–41.
Lee JH, Kim WY, Kim MH and Lee YJ (1993) On the evaluation of Boolean operators in the extended Boolean retrieval framework. In: Proceedings of SIGIR, pp. 291–297.
Lee JH, Kim MH and Lee YJ (1992) Enhancing the fuzzy set model for high quality document rankings. Microprocessing and Microcomputing, 35:337–344.
Meadow CT (1992) Text Information Retrieval System. Academic Press, New York.
Negoita C and Flonder P (1976) On fuzziness in information retrieval. International Journal of Man-Machine Studies, 8:711–716.
Salton G (1989) Automatic Text Processing. Addison-Wesley, Reading, MA.
Sanchez E (1989) Important in knowledge systems. Information Systems, 14:455–464.
Yager RR (1987) A note on weighted queries in information retrieval systems. J. of the American Society of Information Sciences, 38:23–24.
Yager RR (1988) On ordered weighted averaging aggregation operators in multi-criteria decision making. IEEE Transactions on Systems, Man and Cybernetics, 18:183–190.
Yager RR (1993) Families of OWA operators. Fuzzy Sets and Systems, 59:125–148.
Yager RR (1996) Quantifier guided aggregation using OWA operators. International Journal of Intelligent Systems, 11:49–73.
Yager RR (1997) On the inclusion of importances in OWA aggregations. In: Yager RR and Kacprzyk J, Eds., The Ordered Weighted Averaging Operators: Theory and Applications. Kluwer Academic Publishers, Norwell, MA, pp. 41–59.
Yager RR and Kacprzyk J (1997) The Ordered Weighted Averaging Operators: Theory and Applications. Kluwer, Norwell, MA.
Zadeh LA (1983) A computational approach to fuzzy quantifiers in natural languages. Computing and Mathematics with Applications, 9:149–184.
Zadeh LA (1996) Fuzzy logic = computing with words. IEEE Transactions on Fuzzy Systems, 4:103–111.