

# GENERATING AND APPLYING RULES FOR WEB DOCUMENTS RETRIEVAL

Elias Deeba, Andre de Korvin, Ping Chen

*Department of Computer and Mathematical Sciences  
College of Sciences and Technology  
University of Houston-Downtown  
One Main Street  
Houston, TX 77002  
dekorvina, deebae, chenp@uhd.edu*

**Abstract:** Web documents retrieval is very challenging due to the huge amount of documents available and difficulty to interpret these documents. Both effectiveness and efficiency of retrieval are important. This paper presents some approaches from soft computing to improve effectiveness of web documents retrieval. These approaches give a more accurate and reasonable representation of terms provided by the user, present how to match terms and documents with fuzzy logic techniques, and show how to reduce the number of matched documents and necessary terms. A possible architecture for a neuro-fuzzy system to match terms and documents is sketched. This paper also discusses how to form linguistic rules by polling a panel of experts.

**Keywords:** Text retrieval, web mining, soft computing

## 1. INTRODUCTION

Huge amount of data exists in World Wide Web, which makes web mining an interesting and challenging task. Some of the characteristics of the data on the web involves unlabeled data, heterogeneous data, time varying data, distributed data and semi-structured data. Aside information retrieval, some of the challenges are information extraction, pattern recognition, machine learning, validation analysis and knowledge acquisition. In this connection the reader is referred to [6] for a good look at the large scope of web mining extending from text retrieval to multimedia retrieval.

In this paper we propose and discuss some techniques for document retrieval, usually performed by web search engines. Basically there are two phases to find the documents/web pages based on searching criteria provided by the user. First

phase involves matching searching criteria with locally cached documents or documents in the remote web sites. If due to the poor matching quality lots of documents are matched (usually it is the case), we need rank these matched documents according to some non-content criteria (e.g., structural information such as hyperlinks). Some web search algorithms, such as HITS and PageRank, rely on the second phase to rank hundreds of matched documents and present the best fits first to the user. The difficulty of locating highly matched documents comes from:

- Insufficient description of search criteria from the user and,
- Difficulty to interpret the document besides symbol processing.

Our approach uses soft computing techniques and improves effectiveness of documents matching in

the first phase. We present some ideas to reduce the number of terms supplied by the user to describe the documents they want and how to reduce the number of matched documents to improve the matching quality.

Lots of ideas has been proposed for text retrieval. We list some techniques and related problems that we will address in this manuscript:

- (1) *Term-Document Approach* Term relevance is somewhat subjective. The relevance of a term in a document assigned by the designer and the user typically do not match.
- (2) *Training Set Approach* Given that there is only a partial match(i.e. imperfect match), reasonable retrieval methods need to be found. The designed system should have the capability to learn to retrieve better.
- (3) *Document Reduction Approach* Since a very large number of documents exists on the web, it is important to reduce the number of documents relevant to a query.
- (4) *Linguistic Approach* Many users feel more comfortable framing their queries as linguistic queries, e.g. many feel more comfortable stating that relevance of term t is high than stating the relevance of term t is 0.8 ( on a scale of [0,1]).
- (5) *Rule-based Approach* If the capacity of handling linguistic queries is present, that capacity is often provided by a set of rules. How does one obtain such rules given that conflicts of opinions probably will take place, even among experts?
- (6) *Term Reduction Approach* Many terms could be used to describe a document. Therefore, it is important to be able to reduce the number of terms.
- (7) *Neuro-Fuzzy System Approach* Retrieval rules are bound to be fuzzy. Fuzzy rules are only as good as membership functions describing fuzzy terms. How can one learn the membership functions? The designed system should have the capability to learn to retrieve better.

Many researchers in recent works have felt that soft computing seems to be the right approach to handle problems of this nature. The principle tools of soft computing are fuzzy logic, neural networks, genetic algorithms, rough sets and more recently neuro-fuzzy systems.

One of the aims of soft computing is to achieve more "human-like" decisions. For ideas along these lines see [18]. For references to soft computing specifically applied to information retrieval see [9]. For information on linguistic queries see [17]. For applications of fuzzy logic to information retrieval we refer the reader to [11] while for applications of neuro-fuzzy systems to web mining we refer to [10]. Many researchers have used an ar-

tificial intelligence approach, specifically creating agents, for information retrieval, see [12] and [3]. For an application of neural nets to searching see [8] and [7]. Finally for the basic and introductory materials on information retrieval we refer the reader to [4].

Mainly this paper discusses some possible approaches designed to meet the challenges outlined in items 1-7 of this section.

## 2. THE TERM-DOCUMENT MATRIX

A natural setting is to view information retrieval as being defined by a "term-document" matrix. More generally, see [2] for the use of linear algebra in information retrieval.

Let  $d_1, d_2, \dots, d_n$  be  $n$  different documents,  $t = (t_1, t_2, \dots, t_m)^T$  be terms in these documents, and  $\varphi_{ij}$  denotes the degree of certain term  $t_i$  belonging to an document  $d_j$ . Let  $\varphi$  be the  $m$  by  $n$  term-document matrix as below

$$\varphi = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \dots & \varphi_{1n} \\ \varphi_{21} & \varphi_{22} & \dots & \varphi_{2n} \\ \dots & \dots & \dots & \dots \\ \varphi_{m1} & \varphi_{m2} & \dots & \varphi_{mn} \end{bmatrix} \quad (1)$$

This can be translated into the following set of rules:

- If  $t_1 = \varphi_{11}$  and ...  $t_m = \varphi_{m1}$ , then  $I = 1$ ;
- If  $t_1 = \varphi_{12}$  and ...  $t_m = \varphi_{m2}$ , then  $I = 2$ ;
- ...
- If  $t_1 = \varphi_{1n}$  and ...  $t_m = \varphi_{mn}$ , then  $I = n$ ;

Here  $I$  is the index of the relevant documents. That is, if  $t = (t_1, t_2, \dots, t_m)^T$  and  $t_i$  is  $\varphi_{ij}$  relevant to  $d_j$ , then  $d_j$  is the appropriate document and the index  $I$  is  $j$ . We have  $n$  rules, which reflect a possible exact match with a column of matrix  $\varphi$ . Typically, in a query the term vector  $t = (t_1, t_2, \dots, t_m)^T$  may not match the original term  $t = (t_1, t_2, \dots, t_m)^T$ . Therefore, we have to relax the matching by changing the numbers  $\varphi_{ij}$  to a function such as a triangle with a peak at  $\varphi_{ij}$  or a Gaussian function centered at  $\varphi_{ij}$ . Then we have a set of fuzzy rules. Before we do this, we develop formula for computing the index in case we do not have an exact match.

## 3. FORMULA REPORTING INPUT AND OUTPUT

Let  $X = \{x|x \in R_m\}$  and  $x_i$  be the relevance of term  $t_i$ (measured on the same scale as entries of  $\varphi$ ). Then we can define a function  $\varphi_l : X \rightarrow R^+$  by:

$$\varphi_l(x) = \frac{\prod_{k=1}^m \varphi_{kl}(x_k)}{\sum_{p=1}^n \prod_{k=1}^m \varphi_{kp}(x_k)} \quad (2)$$

The meaning of  $\varphi_l(x)$  can be interpreted as the strength normalized to which the  $l^{th}$  rule applies given input  $x$ .

The output can then be approximated by

$$I(x) = \sum_{l=1}^n l\varphi_l(x) \quad (3)$$

$I(x)$  is “the index of the most relevant rule given the term input  $x$ ”. Of course,  $I(x)$  may not be an integer value. In this case we could select the rule which is “the closest” to  $I(x)$ . This would be a reasonable way to pick the right rule provided the documents are “clustered” in the right way. i.e.  $d_s$  is close positionwise to  $d_i$  if their column vectors are close. If the  $d$ 's are not clustered we can pick those  $d$ 's for which  $\varphi_s(x) \geq \tau$ , where  $\tau$  is a predefined number in  $[0,1]$  (i.e., look at those rules whose strength is at least  $\tau$ ).

At any rate, the values of  $\varphi_l(x)$  ( $l=1,2,\dots,n$ ) determine what documents to retrieve. In the next section we consider the case where the documents are not clustered in the right way.

Here is an example to illustrate our method, and the computation is implemented in *Maple9*.

Suppose we generate the following term-document matrix, which has five documents and six keywords used for indexing,

$$A = \begin{bmatrix} 0.8 & 0.2 & 0.1 & 0.9 & 0.1 \\ 0.8 & 0.1 & 0.9 & 0.8 & 0.9 \\ 0.9 & 0.2 & 0.2 & 0.9 & 0.2 \\ 0.1 & 0.1 & 0.1 & 0.8 & 0.2 \\ 0.1 & 0.8 & 0.1 & 0.9 & 0.9 \\ 0.1 & 0.2 & 0.1 & 0.8 & 0.1 \end{bmatrix} \quad (4)$$

Membership functions for these documents are defined as,

$$f_1 = \begin{cases} 0 & , & 0 \leq x \leq 0.6 \\ 5x - 3 & , & 0.6 \leq x \leq 0.8 \\ -5x + 5 & , & 0.8 \leq x < 1 \end{cases}$$

$$f_2 = \begin{cases} 0 & , & 0 \leq x \leq 0.7 \\ 5x - 3.5 & , & 0.7 \leq x \leq 0.9 \\ 1 & , & 0.9 \leq x < 1 \end{cases}$$

$$f_3 = \begin{cases} 1 & , & 0 \leq x \leq 0.1 \\ -5x + 1.5 & , & 0.1 \leq x \leq 0.3 \\ 0 & , & 0.3 \leq x < 1 \end{cases}$$

$$f_4 = \begin{cases} 5x & , & 0 \leq x \leq 0.2 \\ -5x + 2 & , & 0.2 \leq x \leq 0.4 \\ 0 & , & 0.4 \leq x < 1 \end{cases}$$

And with a query,

$$q = \begin{bmatrix} 0.9 \\ 0.8 \\ 0.9 \\ 0.8 \\ 0.9 \\ 0.8 \end{bmatrix} \quad (5)$$

Our program written in *Maple9* shows that the fourth document has a perfect match, i.e.,  $I(a) = 4$ . And in this case the entries 0.1, 0.9, 0.2 and 0.8 of  $A$  were changed into membership functions peaking at these values. More specifically, 0.1 was converted to  $f_3$ , 0.9 was converted to  $f_2$ , 0.2 was converted to  $f_4$ , and 0.8 was converted to  $f_1$ . The linguistic interpretation of  $f_3, f_2, f_4$ , and  $f_1$  is respectively low, high, average low and average high. It should be noted that for more reliable result rules with low strength should be discarded as many such rules could move the index up or down from its legitimate value. In all cases high values of  $\varphi_l(x)$  will be indicators of appropriate documents for  $x$ .

#### 4. USING TRAINING SETS

It may well be the case that one has far more documents than needed for typical queries. We assume here that we have a training set of the form  $(x^1, I^1), \dots, (x^N, I^N)$ , here  $x^i$  is a vector of relevance of terms  $t_1, t_2, \dots, t_m$  and  $I^i$  is the index of the ideal rule to be retrieved for  $x^i$ .

How can we make use of such information? We start by forming:

$$\Phi = \begin{bmatrix} \varphi_1(x^1) & \dots & \varphi_n(x^1) \\ \varphi_1(x^2) & \dots & \varphi_n(x^2) \\ \dots & \dots & \dots \\ \varphi_1(x^N) & \dots & \varphi_n(x^N) \end{bmatrix} \quad (6)$$

Each column of the matrix  $\Phi$  relates to the strength of corresponding rule for each data point  $x^i$  ( $1 \leq i \leq N$ ). We note that  $\varphi_i(x^j)$  denotes the relevance of  $x^j$  to document  $d_i$ .

If the  $d$ 's are clustered in the right way we could have

$$\Phi \begin{bmatrix} 1 \\ 2 \\ \dots \\ n \end{bmatrix} = \begin{bmatrix} \hat{I}^1 \\ \hat{I}^2 \\ \dots \\ \hat{I}^N \end{bmatrix} \quad (7)$$

In the above,  $I^k$  would be the index corresponding to  $x^k$ , and we could have  $\hat{I}^k = I^k$  ( $1 \leq k \leq N$ ) by (3).

What should be done if the  $d$ 's are not clustered in the right way? We need to replace

$$\begin{bmatrix} 1 \\ 2 \\ \dots \\ n \end{bmatrix} \text{ by some predictor } \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_n \end{bmatrix}.$$

So now we set

$$\Phi \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} I^1 \\ I^2 \\ \dots \\ I^N \end{bmatrix} \quad (8)$$

so

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_n \end{bmatrix} = \Phi^+ \begin{bmatrix} I^1 \\ I^2 \\ \dots \\ I^n \end{bmatrix} \quad (9)$$

where  $\Phi^+$  is the pseudo inverse of  $\Phi$ , i.e.,

$$\Phi^+ = (\Phi^T \Phi)^{-1} \Phi^T \quad (10)$$

This yields the least square approximation to the data. The output is now related to the input by

$$I(x) = \sum_{l=1}^n \alpha_l \phi_l(x) \quad (11)$$

where  $\alpha_l$  are given by (9).

Thus the determination of the  $\alpha$ 's using the least squares allows to retrieve the right documents even if they were not originally clustered. The searching for many documents may be computationally expensive. Therefore, we would like to cut down the number of documents and still get "reasonable" matching for a query. To achieve this goal, we want to suggest a method to reduce the rank of the matrix  $\Phi$  which will be discussed in the next section.

Again here we use one example to illustrate our idea. Since using the previous method using this example we could not get a reasonable match, we need to compute the appropriate predictor  $\alpha$ .

Suppose we have a training set as  $(x^l, I^l)$ , ( $1 \leq l \leq N$ ), where

$$x^l = \begin{bmatrix} x_1^l \\ x_2^l \\ x_3^l \\ x_4^l \\ x_5^l \\ x_6^l \end{bmatrix} \quad (12)$$

and  $x^l$  saves the relevance of the 6 terms for input  $x^l$ , and  $I^l$  is the index of right document for input  $x^l$ .

Here the number of documents is 5, and cardinality of training set is also 5. Our training set is

$$x^1 = \begin{bmatrix} 0.9 \\ 0.9 \\ 0.9 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix} \quad (13)$$

and  $I^1 = 1$ .

$$x^2 = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.9 \\ 0.2 \end{bmatrix} \quad (14)$$

and  $I^2 = 2$ .

$$x^3 = \begin{bmatrix} 0.2 \\ 0.8 \\ 0.2 \\ 0.1 \\ 0.1 \\ 0.2 \end{bmatrix} \quad (15)$$

and  $I^3 = 3$ .

$$x^4 = \begin{bmatrix} 0.9 \\ 0.9 \\ 0.9 \\ 0.8 \\ 0.8 \\ 0.8 \end{bmatrix} \quad (16)$$

and  $I^4 = 4$ .

$$x^5 = \begin{bmatrix} 0.2 \\ 0.8 \\ 0.2 \\ 0.2 \\ 0.8 \\ 0.1 \end{bmatrix} \quad (17)$$

and  $I^5 = 5$ .

$$\begin{aligned} \varphi_{11} &= f_1, \varphi_{12} = f_4, \varphi_{13} = f_3, \varphi_{14} = f_2, \varphi_{15} = f_3, \\ \varphi_{21} &= f_1, \varphi_{22} = f_3, \varphi_{23} = f_2, \varphi_{24} = f_1, \varphi_{25} = f_2, \\ \varphi_{31} &= f_2, \varphi_{32} = f_4, \varphi_{33} = f_4, \varphi_{34} = f_2, \varphi_{35} = f_4, \\ \varphi_{41} &= f_3, \varphi_{42} = f_3, \varphi_{43} = f_3, \varphi_{44} = f_1, \varphi_{45} = f_4, \\ \varphi_{51} &= f_3, \varphi_{52} = f_1, \varphi_{53} = f_3, \varphi_{54} = f_2, \varphi_{55} = f_2, \\ \varphi_{61} &= f_3, \varphi_{62} = f_4, \varphi_{63} = f_3, \varphi_{64} = f_1, \varphi_{65} = f_3 \end{aligned}$$

And with a query,

$$q = \begin{bmatrix} 0.8 \\ 0.7 \\ 0.8 \\ 0.7 \\ 0.8 \\ 0.7 \end{bmatrix} \quad (18)$$

We compute

$$I(q) = \alpha_1\varphi_1(q) + \alpha_2\varphi_2(q) + \alpha_3\varphi_3(q) + \alpha_4\varphi_4(q) + \alpha_5\varphi_5(q) = 4.0 \quad (19)$$

which shows that the fourth document is a match.

## 5. REDUCING THE NUMBER OF DOCUMENTS

We use Singular Value Decomposition (SVD) on  $\Phi$ ,

$$\Phi = U\Sigma_r V^T \quad (20)$$

where  $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$ ,  $U$  is  $N \times r$  and  $V^T$  is  $r \times n$  (This is obtained after crossing out appropriate rows of  $V^T$  and columns of  $U$  corresponding to small diagonal values of  $\Sigma$ ). The idea now is to obtain the QR-decomposition of the new matrix  $V^T$ . We review the process for any  $r \times n$  matrix  $H$ . If  $H$  is any  $r \times n$  matrix:

$$H = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ & & \dots & \\ & & & \dots & \dots \end{bmatrix} \quad (21)$$

The QR factorization of  $H$  can be obtained by applying the Gram-Schmidt (G-S) orthogonalization process. For the sake of completeness, we present the G-S algorithm here.

Set  $h_1, h_2, \dots, h_n$  to be the independent columns of  $H$ , thus,

$$q_1 = \frac{h_1}{\|h_1\|} \quad (22)$$

$$q'_i = h_i - P_{i-1}h_i \quad (23)$$

where

$$P_{i-1}h_i = \sum_{k=1}^{i-1} \langle h_i, q'_k \rangle \frac{q'_k}{\|q'_k\|^2} \quad (24)$$

So,

$$H = QR \quad (25)$$

where  $Q$  is a matrix whose columns are the  $q$ 's just constructed and  $R$  is upper triangle. In fact it is easy to verify that the  $p_{th}$  column of  $R$  are the components  $\langle h_p, q_k \rangle$  of  $h_p$  on  $q_1, \dots, q_n$ .

We apply this to  $V^T$ :

$$V^T = \begin{bmatrix} v_1^T \\ v_2^T \\ \dots \\ v_r^T \end{bmatrix} \quad (26)$$

$V^T = QR$ , some columns of  $Q$  are made of 0's. Without loss of generality we may make the

corresponding rows of  $R$  equal to 0 (since such rows are irrelevant in the computation of  $QR$ ).

Move the 0 rows of  $R$  to the bottom of  $R$ . In order to keep the product  $QR$  intact we have to move the corresponding 0 columns of  $Q$  to the right, so we get:

$$Q = \begin{bmatrix} \tilde{q}_1 & \tilde{q}_2 & \dots & \tilde{q}_r & 0 & 0 & \dots & 0 \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot & 0 & 0 & \dots & 0 \end{bmatrix} \quad (27)$$

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \quad (28)$$

Here  $R_{11}$  is upper triangle of size  $r \times r$  and the columns of  $R_{12}$  are linearly dependent on the columns of  $R_{11}$  (the rank of  $R = \text{rank}(V^T) = r$ ).

As an example, let us consider  $V^T$  applied to a query  $X$  as below:

$$V^T X = \begin{bmatrix} a & b & c & a+b+c & a+b-c \\ 0 & d & e & d+e & d-e \\ 0 & 0 & f & f & -f \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \quad (29)$$

In this example, the  $QR$  decomposition of  $V^T$  yields  $Q = I$  and  $R = V^T$ .

The 3 rows of  $RX$  are:

- $a(x_1 + x_4 + x_5) + b(x_2 + x_4 + x_5) + c(x_3 + x_4 - x_5)$
- $d(x_2 + x_4 + x_5) + e(x_3 + x_4 - x_5)$
- $f(x_3 + x_4 - x_5)$

So setting:

- $\bar{x}_1 = x_1 + x_4 + x_5$
- $\bar{x}_2 = x_2 + x_4 + x_5$
- $\bar{x}_3 = x_3 + x_4 - x_5$

The relevant part of  $RX$  in this example is then:

$$V^T = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{bmatrix} \quad (30)$$

In general, the relevant part of  $R$  is then  $R_{11}$  which corresponds to  $\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_r$ . The initial positions of  $\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_r$  then correspond to those columns of  $\Phi$  that should be kept.

We can throw away  $n - r$  documents and still not significantly downgrade the typical information retrieval. We do need to reorganize the remaining term-document matrix.

## 6. LINGUISTIC QUERIES

It is clear that the approach outlined in section 3 can be extended to a linguistic setting. Assume we have a set of rules of the form:

- If  $t_1$  is  $A_{11}$  and ...  $t_m$  is  $A_{m1}$ , then  $I = 1$ ;

- If  $t_2$  is  $A_{12}$  and ...  $t_m$  is  $A_{m2}$ , then  $I = 2$ ;
- ... ..
- If  $t_n$  is  $A_{1n}$  and ...  $t_m$  is  $A_{mn}$ , then  $I = n$ ;

where  $A_{ij}$  are fuzzy sets with membership function  $\phi_{ij}$  and  $I$  is set to the index of the appropriate documents. We have a set of relevance for the  $m$  terms and we would like to obtain the appropriate documents using the rules and the set of relevance may be an  $m$ -dimensional vector or more generally  $m$  fuzzy sets,  $\beta_1, \beta_2, \dots, \beta_m$  where  $\beta_i$  is a linguistic value expressing the relevance of term  $t_i$ . For the  $k^{th}$  rule we compute

$$(\beta_1 \times \beta_2 \dots \beta_m) \square R_k = C_k \quad (31)$$

where  $\square$  denotes the compositional inference:

$$(\beta_1 \times \beta_2 \times \dots \times \beta_m)[\alpha_1, \alpha_2, \dots, \alpha_m] = \beta_1(\alpha_1)t\beta_2(\alpha_2)\dots t\beta_m(\alpha_m) \quad (32)$$

where  $t$  denotes a t-norm. (It is standard to take the max or the min operator for the t-norm).  $R_k$  is a fuzzy relation generated by the  $k^{th}$  rule and is defined by

$$R_k(\alpha_1, \alpha_2, \dots, \alpha_m, i) = A_{1k}(\alpha_1)tA_{2k}(\alpha_2)t\dots tA_{mk}(\alpha_k)tI(i) \quad (33)$$

Of course if we have numerical values for  $I$ , then  $I(k) = 1$  and  $I(i) = 0$  for  $i \neq k$ . We could also make  $I$  fuzzy, i.e., the answer is mostly document  $k$ , however, it could also be document  $h$  or  $j$ , might be translated into:

$$I(k) = 0.9, I(h) = I(j) = 0.4, \text{ and } I(l) = 0 \text{ for } l \notin \{k, h, j\}$$

In fact we want to do this to reduce the number of rules. Again  $t$  in the expression of  $R_k$  denotes a t-norm. The inferential composition is defined by:

$$\begin{aligned} & [(\beta_1 \times \beta_2 \dots \beta_m) \square R_k](i) = \\ & \text{sup}_{\alpha_1, \alpha_2, \dots, \alpha_m} \beta_1(\alpha_1) \wedge \beta_2(\alpha_2) \wedge \dots \wedge \beta_m(\alpha_m) \\ & \wedge R_k(\alpha_1, \alpha_2, \dots, \alpha_m, i) \end{aligned} \quad (34)$$

if we select min for the t-norm.

We denote the result of this operation by  $C_k(i)$ . The result of applying all of the rules leads to:

$$\hat{I}(i) = C_1(i)sC_2(i)s\dots sC_n(i) \quad (35)$$

where  $s$  denotes an s-norm (the standard s-norm used is  $\vee$ , the sup operation). Defuzzifying  $\hat{I}(i)$  and taking the nearest integer to that defuzzification pin-points the appropriate document. An alternate interpretation could be the appropriate documents correspond to those indices  $i$  such that

$\hat{I}(i)$  is close to 1 (or for which  $\hat{I}(i)$  assumes the highest values).

Of course these results will only be as good as the membership functions  $\phi_{ij}$  of  $A_{ij}$ . It is quite possible that some disagreement may be present. For example, with the following relevance indices some would state:

If  $t_1$  is low,  $t_2$  is low,  $t_3$  is high, then the appropriate document is mainly 5 and somewhat 2 and 7 are appropriate.

while others may state:

If  $t_1$  is low,  $t_2$  is low,  $t_3$  is high, then the appropriate document is mainly 10 and somewhat 4 and 13 are appropriate.

In next section we indicate some steps that may be taken to address these issues.

We end this section by pointing out that terms such as low and high may have different meanings even for practitioners of information retrieval. We also point out that once we have rules of the form:

- If  $t_1$  is  $A_{11}$  and ...  $t_m$  is  $A_{m1}$ , then  $I = G_1$ ;
- If  $t_2$  is  $A_{12}$  and ...  $t_m$  is  $A_{m2}$ , then  $I = G_2$ ;
- ... ..
- If  $t_n$  is  $A_{1n}$  and ...  $t_m$  is  $A_{mn}$ , then  $I = G_n$ ;

where  $G_k$  denotes a fuzzy subset of  $\{1, 2, \dots, n\}$  and once the membership functions  $\phi_{kl}$  of  $A_{kl}$  are determined then, as earlier, given a query  $x = [x_1, x_2, \dots, x_m]^T$ , where  $x_i$  denotes the relevance of term  $t_i$  we can form

$$\varphi_l(x) = \frac{\prod_{k=1}^m \varphi_{kl}(x_k)}{\sum_{p=1}^n \prod_{k=1}^m \varphi_{kp}(x_k)} \quad (36)$$

If we use, for example, the "height defuzzification", then, if the documents are clustered tighter in the right way, the closest index to

$$\sum_{l=1}^n \bar{y}_l \varphi_l(x) \quad (37)$$

will index the appropriate document where  $\bar{y}_l$  denotes the value at which  $G_l$  peaks (or the average of the values at which  $G_l$  peaks if  $G_l$  peaks at more than one value). One could then use, as earlier, a least square fit to replace  $\bar{y}_l$  by appropriate  $\alpha_l$  provided there is a training set.

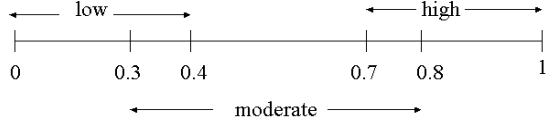
## 7. FORMING THE RULES

Let us suppose, for the sake of specificity, our rules are of the form:

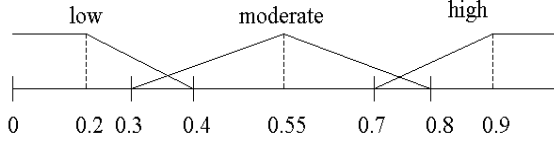
If  $t_1$  is  $A_{1k}$  ... and  $t_m$  is  $A_{mk}$ , then  $I = G_k$ ; ( $1 \leq k \leq n$ )

where the possible values of  $A_{ij}$  are low, medium, high.

We could have a panel expressing their feelings about low, medium, high.



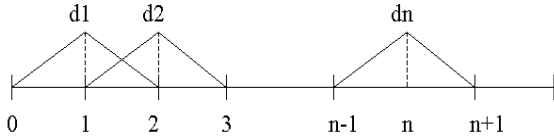
The polling would result in dividing  $[0,1]$  into overlapping regions. In the above figure, 0.4 would be the highest value assigned to low by one member of the panel while 0.3 would be the lowest value assigned to moderate by one member of the panel. We would then generate the following membership functions:



Here 0.2 could be, for example, the highest score assigned to low by 80 percent of the panel. A similar interpretation would hold for 0.9. We now can form the rules using a training set. A typical training set is of the form:

If  $t_1 = f_1$  and ...  $t_m = f_m$ , then  $I = g$

where  $f_i \in [0,1]$  ( $1 \leq i \leq m$ ) and  $g \in [1, n]$ . If  $g = 3.2$ , for example, it would indicate the extent to which document 3 is preferred to document 4 as the ideal answer to the values of  $t$ . Of course  $f_i$  is the relevance of term  $t_i$ . It is understood that  $[1, n]$  is partitioned in a manner similar to the terms, e.g.



The training set should be well chosen so that the  $I$  term covers uniformly the range of documents. Each point in the training set generates one rule. For example, if  $t_1 = 0.25, t_2 = 0.42, t_3 = 0.85, I = 3.2$ , then the rule "If  $t_1$  is low,  $t_2$  is moderate and  $t_3$  is high, then the ideal document is 3" would be generated. Very likely some conflicting rules would be generated, i.e., rules with the same antecedent and different consequents. An example of a rule conflicting with the above rule would be

If  $t_1 = 0.1, t_2 = 0.6, t_3 = 0.95, I = 2.4$ ,

Then the rule generated would be:

If  $t_1$  is low,  $t_2$  is moderate and  $t_3$  is high, then the ideal document is 2.

The rule with the highest applicability should take precedence. The degree of applicability of the two rules are:

- $\text{low}(0.25) \wedge \text{moderate}(0.42) \wedge \text{high}(0.85) \wedge d_3(3.2)$
- $\text{low}(0.1) \wedge \text{moderate}(0.6) \wedge \text{high}(0.95) \wedge d_2(2.4)$

Once we have the rules, information retrieval proceeds as described earlier. The SVD decomposition can be used to reduce the number of rules.

## 8. REDUCING THE NUMBER OF TERMS

It is important to select terms that have more discriminating power. For example, if the relevance of  $t_1$  is approximately 0.3 for all documents,  $t_1$  does not contribute much to the retrieval of documents. The methodology known as the principal component analysis seeks to linearly combine terms in such a way that the variance of the projection on these linear combinations is maximized. For additional information on that method see [1], [5] and [15]. Assume that we have a set of terms with special values. We then have a sequence  $x_1, x_2, \dots, x_i, \dots$  of  $m$ -dimensional vectors where

$$x_i = [t_1^i, t_2^i, \dots, t_m^i]^T \quad (38)$$

where  $t_j^i$  denotes the value of term  $t_j$  for the  $i^{\text{th}}$  query.

We seek an  $m$ -dimensional unit vector  $u$ , so that the variance of the projections of  $x_i$  on  $u$  is maximal. Of course  $u$  will be considered a new term vector obtained by an appropriate linear combination of the term vectors  $x_i$ . The projections of  $x_i - \bar{x}$ , where  $\bar{x}$  is the mean vector are given by:

$$\begin{aligned} p_i &= \langle x_i - \bar{x}, u \rangle \\ &= (x_i - \bar{x})^T u \\ &= u^T (x_i - \bar{x}) \end{aligned} \quad (39)$$

then

$$\begin{aligned} \sum_i p_i &= \sum_i u^T (x_i - \bar{x}) \\ &= u^T \sum_i (x_i - \bar{x}) \\ &= u^T 0 \\ &= 0 \end{aligned} \quad (40)$$

$$\begin{aligned} p_i^2 &= u^T (x_i - \bar{x})(x_i - \bar{x})^T u \\ &= u^T [(x_i - \bar{x})(x_i - \bar{x})^T] u \end{aligned} \quad (41)$$

so the variance of the projections is:

$$\begin{aligned} \frac{1}{n} \sum_i p_i^2 &= u^T \left[ \frac{1}{n} \sum (x_i - \bar{x})(x_i - \bar{x})^T \right] u \\ &= u^T R u \end{aligned} \quad (42)$$

where R is the correlation matrix  $E[x - \bar{x}][y - \bar{y}]$ .

To maximize this, we construct the Lagrangian functional with  $\lambda$  being the Lagrange multiplier:

$$J = u^T R u + \lambda(1 - u^T u) \quad (43)$$

To maximize J, we compute the gradient of J and set it equal to 0. This yields

$$\nabla_u(J) = 2Ru - 2\lambda u = 0 \quad (44)$$

So to maximize the variance  $u^T R u$  of the projections, we pick u so that  $Ru = \lambda u$ . Since

$$u^T R u = u^T \lambda u = \lambda \quad (45)$$

we need to pick u to be the eigenvector of R with the largest possible eigenvalue  $\lambda$ .

More generally, if one orders the eigenvalues of R as

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N \quad (46)$$

and picks  $\lambda_1, \lambda_2, \dots, \lambda_r$ , then the eigenvectors  $u_1, u_2, \dots, u_r$  become the new term vectors. Each query  $q = [h_1, h_2, \dots, h_m]$  is then expressed as  $q' = [q_1, q_2, \dots, q_r]$ , where  $q' = \sum_{i=1}^r q_i u_i$  and query involving m terms are converted into queries involving r terms.

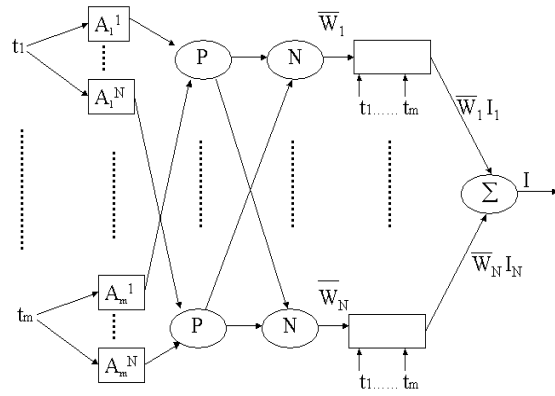
## 9. REFINING THE MEMBERSHIP FUNCTIONS

Of course the rules are only as good as the membership functions of the fuzzy terms involved. In this section we indicate a possible approach to this problem. We may approximate rules discussed in the previous sections by a Sugeno fuzzy inference system(see [14]). Such a system has rules of the form:

$R_k$ : if  $t_1$  is  $A_1^k$ ,  $t_2$  is  $A_2^k$ , ... and  $t_m$  is  $A_m^k$ , then  $I_k = \sum_{j=1}^m p_j^k t_j + r_k$

In other words the appropriate document index is given by some linear functions of the terms. Intuitively our former rules can be approximated by such rules because most functions can be approximated by linear functions locally.

We start the discussion by requiring that only one document is the ideal answer to a query and then we generalize this allowing a set of documents to be the answer to a query. A possible architecture to provide an answer to rules of the form  $R_k$  above is given in the following diagram.



The square nodes denote nodes containing a parameter. For example, the  $A_h^l$  ( $1 \leq l \leq N, 1 \leq h \leq m$ ), where N denotes the number of rules) denotes linguistic terms such as high, low, etc.. The boxes with input  $t_1, t_2, \dots, t_m$  denote nodes with parameters  $p_1^k, p_2^k, \dots, p_m^k, r_k$ .

The nodes P output the corresponding products, e.g.  $\prod_{h=1}^m A_h^l(t_h)$ . The nodes labeled with N output the normalized products, e.g.,

$$\bar{W}_l = \frac{\prod_{h=1}^m A_h^l(t_h)}{\left( \sum_{l=1}^N \prod_{h=1}^m A_h^l(t_h) \right)} \quad (47)$$

Finally the box labeled  $\Sigma$  outputs the weighted sum of  $I_k$ , i.e.

$$I = \sum_{l=1}^N \bar{W}_l I_l \quad (48)$$

Several methods can be used to refine the values of the parameters through the use of a training set. We briefly outline one of the methods known as the Hybrid Learning Algorithm. The training set consists of tuples of the form

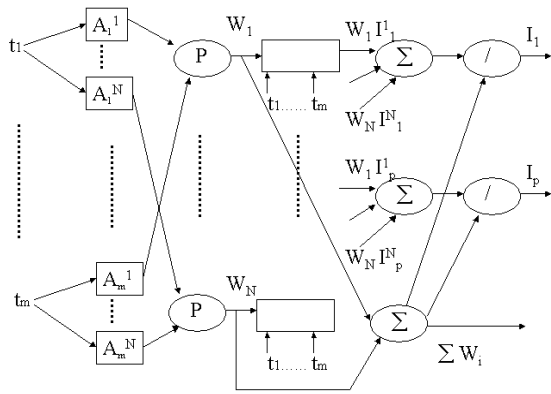
$$(t_1^j, t_2^j, \dots, t_m^j, I^j), \quad j = 1, 2, \dots, m$$

where  $I^j$  is the index of the document that best answers the query  $(t_1^j, t_2^j, \dots, t_m^j)$ . Since I is a linear function of  $p_1^k, p_2^k, \dots, p_m^k, r_k$ , ( $1 \leq k \leq n$ ), a linear square fit can be used to obtain a value for these parameters. Then taking another training set a forward pass is performed and the result is compared with the target result. The error is then backpropagated to obtain the appropriate adjustment on the parameters of  $A_l^h$ . In fact one can perform a sequence of least square fit followed by backpropagation. We finally sketch a possible architecture that would adjust rules of the form

$R_k$ : if  $t_1$  is  $A_1^k$ ,  $t_2$  is  $A_2^k$ , ... and  $t_m$  is  $A_m^k$ , then

$$I_1^k \text{ and } I_2^k \dots \text{ and } I_p^k$$

These rules yield the p best documents for a given query.



Clearly the Hybrid Learning Algorithm can be applied to this setting. For backpropagation in the context previously discussed see [16] and for architecture pertaining to Sugeno Fuzzy Systems we refer to [14] and [15].

## 10. CONCLUSION

We have briefly outlined some possible approaches to address the problems outlined in 1-8 of the introduction section. In section 2 we presented the term-document matrix and extended the entries to be fuzzy sets to allow for partial matching. In section 3 we gave a formula to obtain the right document. In section 4 we sought to improve that input-output formula through the use of a training set and the use of the least square fit. In section 5 we indicated how the use of SVD decomposition can reduce the number of documents of linguistic queries. In section 6 we initiated a discussion of linguistic queries and in section 7 we indicated how linguistic rules might be formed by polling a panel of experts. We showed how the use of principal component analysis may reduce the number of terms. Finally we sketched some possible neuro-fuzzy architecture that could be used to refine the membership functions of the fuzzy rules. It seems clear that soft computing can and will play a significant role in web mining.

## REFERENCES

- [1] T.W. Anderson, An introduction to multivariate statistical analysis. John Wiley and Sons, N.Y. 1984
- [2] M.J.A. Berry, Z. Drmvac and E.R. Jessup, Matrices, Vector Spaces and Information Retrieval. SIAM Review, 41(2) (1999) 335-362
- [3] H. Chen, M. Ramsey, and P. Li, The Java search agent workshop, Soft Computing in Information Retrieval: Techniques and Applications. Vol 50. Heidelberg: Physica Verlag (2000) 122-140
- [4] W. B. Frakes and R. Baeza-yates (ed) Information Retrieval: Data Structures and Algorithms. Englewood Cliffs, NJ. Prentice Hall 1992.
- [5] T. Hoffmann, Probabilistic Latent Semantic Indexing. Proc. of the ACM SIGIR Conf. NY ACM Press(1999) 50-57.
- [6] J.H. Lim Visual keywords: from text retrieval to multimedia retrievals. Soft Computing in Information Retrieval: Techniques and Applications. Vol 50. Heidelberg: Physica Verlag (2000) 77-101
- [7] C.T. Lin and C.S.G. Lee, Neural network based fuzzy logic control and decision system. IEEE Transactions on Computers 40(12)(1991) 1320-1336.
- [8] D. Merkl and A. Rauber Document classification with unsupervised artificial neural networks. Soft Computing in Information Retrieval: Techniques and Applications. Vol 50. Heidelberg: Physica Verlag (2000) 102-121
- [9] S. Mitra, S. K. Pal and P. Mitra. Data mining in soft computing framework: a survey. IEEE Transactions on Neural Networks, to appear.
- [10] S.K. Pal and S. Mitra, Neural Fuzzy Pattern Recognition: Method of Soft Computing. John Wiley and Sons, N.Y., 1999.
- [11] G. Pasi, G. Bordonga, Applications of Fuzzy Set Theory to Extend Boolean Information Retrieval. Soft Computing in Information Retrieval: Techniques and Applications. Vol 50. Heidelberg: Physica Verlag 2000 21-47.
- [12] J. Shavlik and T. Eliassi. A system for building intelligent agents that learn to retrieve and extract information. International Journal on User Modeling and User Adopted Interaction 2001.
- [13] G.A.F. Seber. Multivariate Observations. John Wiley and Sons, N.Y. 1984.
- [14] M. Sugeno and G. T. Kang, Structure Identification of Fuzzy Model. Fuzzy Set and Systems, 28(1988) 15-33
- [15] T. Takagi and M. Sugeno. Fuzzy Identification of Systems and its Applications to Modelling and Control". IEEE Transactions on Systems, Man and Cybernetics 15, 1985 116-132.
- [16] L. X. Wang and J. M. Mendel, Backpropagation Fuzzy Systems As Nonlinear Dynamic System Identifiers. Proc. of IEEE International Conference on Fuzzy Systems, San Diego 1992
- [17] R. Yager, A Framework for Linguistic and Hierarchical Queries for Document Retrieval. Soft Computing in Information Retrieval: Techniques and Applications. Vol 50. Heidelberg: Physica Verlag (2000) 3-20
- [18] L. A. Zadeh. Fuzzy Logic, Neural Networks and Soft Computing. Communications of ACM, vol 37(1994) 77-84