International Journal of Pure and Applied Mathematics
Volume 44 No. 1 2008, 51-62

INFORMATION RETRIEVAL USING RELEVANCE VECTORS: A SOFT COMPUTING APPROACH

Andre de Korvin[1] §, Ping Chen[2], Jeong-Mi Yoon[3]
[1,2,3]Department of Computer and Mathematical Sciences
University of Houston-Downtown
One Main Street – RM. S705, Houston, TX 77002, USA
[1]e-mail: korvinA@uhd.edu
[2]e-mail: chenp@uhd.edu
[3]e-mail: yoonj@uhd.edu

Abstract: In this paper we propose to design a document retrieval system based on the degree of relevance of a set of terms to the documents. The set of relevance of terms has been divided among a number of expert systems. Each system forms its own set of rules based on CART and subsequent pruning of the tree. The method is observed so that each rule pertains to a number of documents whose average relevance is close to 1. These rules are then fuzzified to take care of boundaries separating the rules. We then propose a possible network to implement the document retrieval and the hybrid learning method to train this net. The rules we obtained are of the Sugeno type and the net is an ANFIS construct. Finally we proposed a method to aggregate the decisions of the different expert systems by using a hierarchical structure involving a gating network.

AMS Subject Classification: 68P20
Key Words: information retrieval, soft computing

1. Introduction

Standard indexing for information retrieval may involve the use of titles, author's names, keywords and subject classification. The problem faced is an even increasing volume of information as well as conflicting methods of classifying

Received: January 26, 2008

§Correspondence author

materials. Roughly 12,000 periodicals are added to the pool each year. Approximately 60,000 new book titles appear each year. There are approximately 300 million web pages on the Internet. For extensive information on such statistics we refer the reader to [2].

In the interesting article by M. Berry, Z. Drmac and E.R. Jessup it is shown how linear algebra can be used to manage and index large text coefficients [1]. In this work, each document is encoded as a vector where each component reflect the importance of a particular term in representing the semantics of that document. The SMART system (System for the Mechanical Analysis and Retrieval of Text) is one of the first system to use the vector space approach [5]. In [1] it is pointed out that the cosine of the angle between a query and document vectors is a viable measure of similarity, i.e.,

$$\cos\theta_j = \frac{< a_j, q >}{||a_j|_2||q||_2}\,, \tag{1}$$

where a_j and q denote the document vector j and the query q vector $< a_j, q >$ denotes their inner product and $||.||_2$ denotes the usual norm.

In [16] Fuzzy Rules are generated and used for web document retrieval. Techniques of "Soft Computing" are applied to achieve more human-like decisions. For works along these lines, we refer the reader to [6], [12], [13]. The methodologies typically used involve fuzzy logic, neural nets, neuro-fuzzy systems and intelligent agents.

In [2], the authors present paradigms for decision making under increasing levels of uncertainty. While information retrieval per-say is not treated, some of the methods are potentially applicable to the present situation.

We now outline the purpose of the sections following this introduction. In Section 2, using CART [4] (Classification and Regression Tree) we construct rules of the form:

If t_1^i is $A_{1,k}^i$ and t_2^i is $A_{2,k}^i$ and ... $t_{N_i}^i$ is $A_{N_i,k}^i$, then

$$Z_k = \sum_{l=1}^{N_i} \alpha_{l,k} t_l^i\,. \tag{2}$$

The above rule refers to the k-th rule generated by expert i whose expertise focuses on N_i terms whose relevance are t_l^i $(1 \le l \le N_i)$. The $A_{j,k}^i$ $(1 \le j \le N_i)$ are subsets, in fact subintervals, of $[0, 1]$ as we assume the relevance of each term is a scalar in $[0, 1]$. The consequent Z_k is the average similarity (which could be taken to be the average cosine) of documents fitting the antecedent of the rule.

In Section 3, we proceed to reduce the number of rules obtained in the previous section. Even though CART was used the number of rules obtained may still be too large. We apply an algorithm used by L. Breiman, J.M. Friedman, R.A. Olishen and C.J. Stone [4] to reduce the number of rules. The main idea there is to introduce the concept of "cost-complexity".

In Section 4, we "smooth-out" the rules. The $A^i_{j,k}$ obtained previously are disjoint intervals, we change these to fuzzy sets, to better take care of boundary situations.

In Section 5, we present the ANFIS architecture (Adaptive Neural Fuzzy Inference System) to implement the Sugeno Fuzzy Rules, which are precisely the type of rules constructed in our previous sections. For additional background on Sugeno Fuzzy Systems and ANFIS architectures we refer the reader to [17], [18], [19], [8].

In Section 6, we train the ANFIS net using the hybrid learning method. This method uses a least square fit to tune the parameters present in the consequent and uses a backpropagation to tune the parameters present in the antecedent.

In the final Section 7, we put together the decisions made by each expert i, introducing additional parameters g_i reflecting the credibility of expert i. This is done by using the methodologies developed in [9] and [15]. Essentially two nets are used, the experts nets and gating nets.

2. Obtaining Crisp Rules

Following the approach of [1] we encode documents and queries as vectors where each component reflects the importance of a particular term in representing the semantics of documents/queries. We assume the existence of a "set of experts" where each "expert" is knowledgeable about a set of terms.

We denote the i-th such expert by expert i. Thus, each expert i deals with components $t^i_1, t^i_2, ..., t^i_{N_i}$ of the semantics of a document. Two different experts, expert i and expert j may have overlapping components. The goal in this section is to form rules of the form.

If t^i_1 is $A^i_{1,k}$ and ..., $t^i_{N_i}$ is $A^i_{N_{1,k}}$, then

$$Z^i_k \approx \sum_{l=1}^{N_i} \alpha^i_{l,k} t^i_l \,. \tag{3}$$

The index k refers to the k-th rule formed by expert i. The sets $A^i_{j,k}$ $(1 \leq$

$j \leq N_i$) are appropriate intervals. The intuitive meaning of Z_k^i is the "average similarity" of documents fitting that k-th rule. According to [1] the similarity of two documents may be taken to be the cosine of their angle but we prefer to use the more general concept of similarity in the present work.

Without loss of generality we assume that all components are normalized, so $t_{j,k}^i$ is in $[0,1]$ and it will follow that the $A_{j,k}^i$ will be appropriate subintervals of $[0,1]$.

To construct the $A_{j,k}^i$ we will use the CART method [4]. This involves building a tree and defining an error as follows:

$$E(n) = Min \sum_{p=1}^{N(n)} (y_p - \sum_{l=1}^{N_i} \alpha_l^i t_{l,p}^i)^2 \,. \tag{4}$$

$N(n)$ denotes the cardinality of the training set for node n. A typical element of the training set is the pair,

$$\left(\left[\begin{array}{c} t_{1,p}^i \\ \cdots \\ t_{N_i,p}^i \end{array} \right], y_p \right) ,$$

where y_p, denoting the average similarity, are scalars close to 1 and $[t_{1p}^i, ..., t_{N_i,p}^i]^T$ denotes a vector chosen in the node n. We assume there is a number of "possible splits" for each component $t_l^i \in [0,1]$. That is we may choose $t_l^i \leq s_{k,l}^i$ or $t_l^i > s_{k,l}^i (1 \leq k \leq M_l^i)$. That is, we have M_l^i possible choices on how to split the l-th component of each stage k. Note that M_l^i could depend on k, for each of notation we do not introduce the index k in M_l^i.

We now consider the quantity

$$\triangle E(n, s_{k,l}) = E(n) - (E_{p_k,l}) - Eq_k, l) \,, \tag{5}$$

where $E_{p_k,l}$ is obtained from $E(n)$ by looking at documents in the node n consisting only of those documents satisfying $t_l^i \leq s_{k,l}$ and $E(q_{k,l})$ consisting only of those documents satisfying $t_l^i > s_{k,l}$. $\triangle E$ denotes of course the corresponding decrease in error if n is replaced by its left and right sons $p_{k,l}$ and $q_{k,l}$. We pick that split $s_{k,l}$ that maximizes the decrease in error $\triangle E(n, s_{k,l})$.

The root of the tree consists of documents whose terms fall under the expertise of expert i. One then proceeds to construct the binary tree by replacing each node by its two sons. One stops when the errors at the leaves fall below some threshold. The coefficients α_l^i are then computed by a least square fit at the leaves.

At this point the documents containing the t_l^i have been split into disjoint

classes and the $A^i_{j,k}$ form in fact subintervals of $[0,1]$. Each leaf of the tree corresponds to one rule. Each Z^i_k is close to 1, the average relevance of documents covered by the k-th rule.

3. Reducing the Number of Rules

The tree obtained in the previous section may need to be pruned because the previous process may "overfit" the data. A method developed in [4] considers the "cost-complexity" of the tree. The complexity of a tree is defined as $\sum_w E(n^i_w)$ where n^i_w denotes a leaf of the tree (relative to expert i of course). Then the cost-complexity is defined as,

$$E_\alpha(T) = \sum_w E(n^i_w) + \alpha|L|\,, \tag{6}$$

where $|L|$ = number of leaves of T and α is some parameter.

Algorithm. (see [4])

1. For each interval node of T set

$$\alpha_t = \frac{E(T) - E(T_t)}{|\tilde{T}_t| - 1}\,, \tag{7}$$

where T_t is the tree rooted at node t and $|\tilde{T}_t|$ denotes the number of leaves of T_t.

2. Find the minimum of α_t over all nodes t and pick $T - T_t$ as the next minimizing tree. Here $T - T_t$ denotes the tree obtained from T by deleting all nodes of T_t except its root t (thus t becomes a leaf of $T - T_t$).

Repeat the above process until the original tree T is reduced to one node.

We now have a sequence of trees

$$\{t_1\} = T_1 \subset T_2 \subset ... \subset T_L = T. \tag{8}$$

The tree T_i has i nodes (at each stage a pair of leaves is replaced by the parent of these leaves).

Now one can pick an independent test data and select that tree T_i that yields the minimum error on that independent data set.

At this point one has a reduced number of crisp rules of the form as equation (3).

Presumably Z_k is fairly close to 1 and denotes the average similarity of documents defined the above k-th rule.

4. Smoothing Out the Rules

The purpose of this section is to get rid of possible discontinuities at the boundaries of the splits. To accomplish this we may represent $t_l^i \geq c$ by the function

$$A_{c,l}^i = \frac{1}{1 + \exp[-\alpha(y - c)]}. \tag{9}$$

Thus $A_{c,l}^i$ is a function of y with parameter α. Of course $t_l^i < c$ will be represented by $1 - A_{c,l}^i$. Now the k-th rule of expert i is of the form of equation (1), where $A_{j,k}^i$ are appropriate fuzzy sets and Z_k is a scalar (close to 1).

Note. An interval of the form $a_l^i \leq T_l^i \leq b_l^i$ is then transformed into a fuzzy set of the form $A_{a,l}^i t(1 - A_{b,l}^i)$, where t is a t-norm and $a = a_l^i$, $b = b_l^i$ (common t-norms are products or min operators).

The parameters to be determined are:

(1) The α_l^k of the consequents.

(2) The α and c of the antecedents.

In the following section we sketch a possible network to determine the above parameters.

The above rules form a Sugeno Inference System and the network will have the ANFIS architecture (Adaptive Neural Fuzzy Inference System). For additional information on these topics, the reader is referred to [8], [17], [18], [19].

5. The ANFIS Architecture

An ANFIS netork consists of two parts. The first part deal with linguistic rules, the second part is denoted by "The Neural Rules". In our case the linguistic part will involve the fuzzy sets $A_{j,k}^i (1 \leq j \leq N_i)$ and the neural part will use α_l^k as weights. In the present work we present two versions of the neural part. The neural part will be used to of course tune the α_l^k scalars while the linguistic part will be used to tune the α and c parameters of the antecedent.

We have a training set $\left(\begin{bmatrix} t_{1,p}^i \\ \dots \\ t_{N_i,p}^i \end{bmatrix}, y_p \right) (1 \leq p \leq M_i)$.

$\hat{y_p}$ will be a scalar or a vector, depending which version is adapted for the neural part. Of course $[t_{1,p}, ..., t_{N_i,p}]^T$ is a query. Each j component of the

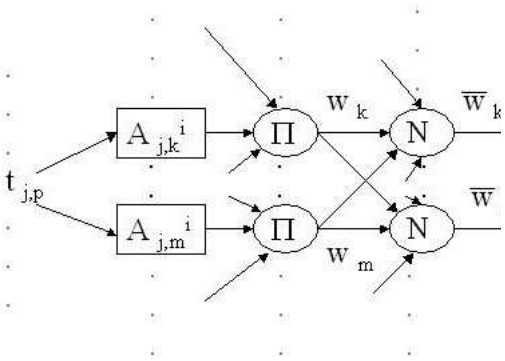

Figure 1: Linguistic part

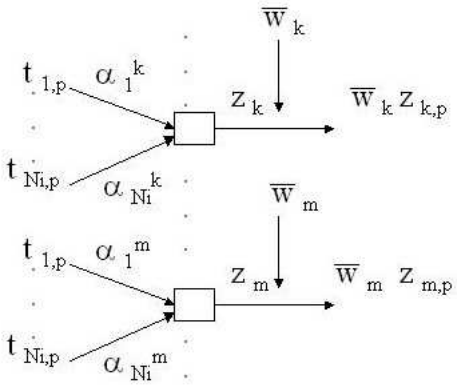

Figure 2: Neural part. Version 1

training set connects to the j-th component of each rule by which expert i operates.

The layer π_i involves taking the appropriate t-norm of terms in the antecedents within the same rule (i.e., the t-norm of the terms $A^i_{1,k}(t_{1,p}), .., A^i_{N_i,k}(t_{N_i,p})$). As mentioned earlier such a t-norm is often a product or the minimum operator.

The layer N performs normalization. Thus the output is the normalized strength of each rule defined by expert i. The first version of the neural part is shown in Figure 2.

The output of the linguistic part is channeled to the second part as shown. Thus the final output is $\bar{w_1}Z_{1,p}..., \bar{w_k}Z_{k,p}, ...$

What is the desired output? The k-th component should have target $\hat{y_{k,p}}$- which should be the average similarity of the query with documents covered by rule k. Thus to obtain $\hat{y_{k,p}}$ we take a sample of documents covered by rule k and average out their similarity (e.g., cosine) with the query $[t_{1,p}, ..., t_{N_i,p}]^T$.

The second version of the neural part is shown in Figure 3.

In this case $\hat{y_p}$, the target, should be the averaged similarity of the query $[t_{1,p}, ..., t_{N_i,p}]^T$ with the strongest rule k.

In the next section we outline a method to train the above ANFIS network.

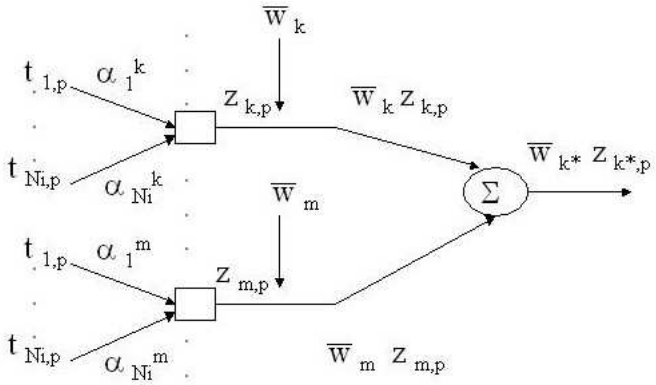

Figure 3: Neural part. Version 2

6. Training The ANFIS Network

Several methods are available to train our ANFIS network. We focus the training on the first version and focus on a convenient method, the hybrid method. In the first version the output is $\bar{w_k} \sum_{j=1}^{N_i} \alpha_j^k t_{j,p}$ for the p-th element of the training set. Let R_i denote the number of rules for expert i, thus $1 \leq k \leq R_t$. The target is $\hat{y_{p,k}}$ defined in the previous section.

We would like the output to match the target, i.e.,

$$\sum_{j=1}^{N_i} \bar{w_k} t_{j,p} \alpha_j^k = \hat{y_{p,k}}, \quad 1 \leq p \leq M_i, \quad 1 \leq k \leq R_i. \tag{10}$$

Or written differently:

$$A_k \begin{bmatrix} \alpha_1^k \\ ... \\ \alpha_{N_i}^k \end{bmatrix} = \begin{bmatrix} \hat{y_{1,k}} \\ ... \\ \hat{y_{M_i,k}} \end{bmatrix}, \tag{11}$$

where the l-th row of A_k is $[\bar{w_k} t_{1,l}, ..., \bar{w_k} t_{N_i,l}]$. Thus A_k is an M_i x R_i matrix.

We need to have $M_i \geq R_i$. Then

$$\begin{bmatrix} \alpha_1^k \\ ... \\ \alpha_{N_i}^k \end{bmatrix} = A_k^+ \begin{bmatrix} \hat{y_{1,k}} \\ ... \\ \hat{y_{M_i,k}} \end{bmatrix}, \tag{12}$$

where A_k^+ is the pseudo-inverse of A_k, i.e., $A_k^+ = (A_k^T A_k)^{-1}$.

Once the α_l^k are determined, back propagation can be used to determine

the parameters of the antecedents (i.e., the α and c parameters). Of course one could repeatedly alternate between the least square fit and back propagation to further tune the parameters.

Note. Should $A_k^T A_k$ not have an inverse it can be replaced by $A_k^T A_k + \lambda I$ ($\lambda > 0$, I – the identity parameter).

7. Aggregating the Experts

Up to now we have considered the decision of a typical expert, i.e., expert i. In this section we would like to form a decision reflecting the aggregation of all experts involved. Recall that each expert deals with its own set of relevance terms and that some terms may possibly overlap across expertise.

A possible way to accomplish the aggregation is to use a version of the methodologies outlined in [15]. In this work the outputs of simple problems are combined stochastically to obtain a global solution.

This particular model has two basic components: the gating networks and the expert networks (the networks corresponding to expert i in our case). The expert networks located at the bottom of the tree receive their inputs (relevance terms in our case) and produce an estimate of the output. The gating networks also receive the input and produce output that weights the contributions of the child networks.

In [15] it is proposed to train the model top-down although the direction. Top-down or bottom-up depend on the problem under consideration.

A possible architecture for our setting is sketched in Figure 4 using the first version of the neural rules. A similar net can be constructed for the second version.

The vector $[t_1, ..., t_m]^T$ denotes a particular query. The scalar r_k^i denotes the strength of the k-th rule according to expert i, relative to the query given. The output given by the vector of component r_k^i denote the rules to be applied to the particular query coming from all of the experts.

The r_k^i denote only the relevant rules to the particular query (the irrelevant rules have their corresponding r_k^i set to 0).

The gating network can, as in [15], be trained so that $g_k^i = 1$ for relevant rules (i.e., for those rules whose strength exceeds some threshold) and $g_k^i = 0$ for irrelevant rules to the query. Thus for a given query, the system will output the most relevant rules coming from the different experts.

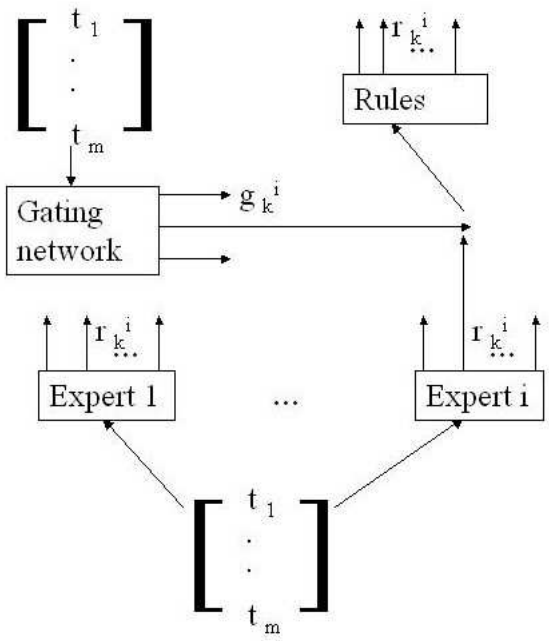

Figure 4: Hierarchical organization for expert aggregation

A collection of documents satisfying these relevant rules may then be determined in a number of ways (perhaps the easiest way would be to return a sample of documents formed by taking the union of samples determined by each individual rules).

8. Conclusion

A number of steps has been proposed in this work to design a possible system to retrieve documents based on the degree of relevance of a set of terms to the documents. The set of relevance of terms has been divided among a number of expert systems. Each system forms its own set of rules based on CART and subsequent pruning of the tree. The method is observed so that each rule pertains to a number of documents whose average relevance is close to 1. These rules are then fuzzified to take care of boundaries separating the rules. We then proposed a possible network to implement this and the hybrid learning method to train this net. The rules we obtained are of the Sugeno type and

the net is an ANFIS construct. Finally we proposed a method to aggregate the decisions of the different expert systems by using a hierarchical structure involving a gating network. It is clear that much research remains to be done on some of the possibilities raised in this article. In fact, different versions of what has been outlined in each section may be constructed and analyzed.

References

[1] M.W. Berry, Survey of public domain lanczos-based software, In: *Proc. of the Cornelius Lanczos Centenary Conf.* (Ed-s: J. Brown, M. Chu, D. Ellison, R. Plemmons), SIAM, Philadelphia, PA (1997), 332-334.

[2] M.W. Berry, Z. Drmac, E. Jessup, Matrices, vector spaces and information retrieval, *Siam Review*, **41**, No. 2 (1999), 335-362.

[3] D. Bogard, Ed., *The Bowker Annual Library and Book Trade Almanac*, 43-rd Edition, R.R. Bowker, New Providence, N.Y. (1998).

[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees*, Belmont, CA Wadsworth International Group (1984).

[5] C. Buckley, G. Salton, J. Allan, A. Singhani, Automatic query expansion using SMART: TREI 3, In: *Overview of the 3-rd Text Retrieval Conf.* (Ed. D. Harman), National Institute of Standards and Technology Special Publication, 500-226, NIST Gaithersburg, MD (April 1995), 69-80.

[6] H. Chen, M. Ramsey, P. Li, The *Java* search age soft computing in information retrieval: Techniques and applications, *Physica*, Verlag, **50** (2000), 122-140.

[7] E. Deeba, A. de Korvin, P. Chen, Generating and applying rules for web document retrieval, *Far East Journal of Applied Mathematics*, **16**, No. 3 (2004), 249-272,

[8] J.S.R. Jang, ANFIS: Adaptive-network-based fuzzy inference systems, *IEEE Trans. on Systems, Man. and Cybernetics*, **23** (1993), 665-685.

[9] M.I. Jordan, R.A. Jacobs, *Hierarchical Mixtures of Experts and the EM Algorithms*, Technical Report, A.I. memo No. 1440, Massachusetts Institute of Technology (1993).

[10] A. de Korvin, F. Modave, R. Kleyle, Paradigms for decision making under increasing levels of uncertainty, *International Journal of Pure and Applied Mathematics*, **21**, No. 4 (2005), 419-430.

[11] R.S. Lawrence, C. Gifes, Searching the World-Wide Web, *Science*, **280** (1998), 98-100.

[12] C.T. Lin, C.S.G. Lee, Neural network based fuzzy, *Control and Decision System IEEE Transaction of Computer*, **40**, No. 12 (1991), 1320-1336.

[13] S. Mitra, S.K. Pal, P. Mitra, Data mining in soft computing framework: A survey, *IEEE Transaction of Neural Network*, To Appear.

[14] G. Pasi, G. Bordonga, Applications of fuzzy set theory to extend Boolean information retrieval, *Soft Computing in Information Retrieval: Techniques and Applications*, Physica Verlag, **50** (2000), 21-47.

[15] M.E. Ruiz, P. Srinivasan, *Hierarchical Text Categorization Using Neural Networks*, Kluwer Academic Publishers (2002), 1-28.

[16] G. Salton, M. Mcgill, *Introduction to Mordern Information Retrieval*, Mcgraw-Hill, New-York (1983).

[17] M. Sugeno, G.T. Kang, Structure identification of fuzzy model, *Fuzzy Sets and Systems*, **28** (1998), 15-33.

[18] T. Takagi, M. Sugeno, Deviation of fuzzy control rules from human operator's control actions, In: *Proceeding of the IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis* (1983), 55-60,

[19] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE Transaction on Systems, Man. and Cybernetics*, **15** (1985), 116-132.