Adaptive Concept-based Retrieval Using a Neural Network*

Minkoo Kim+ and Vijay V. Raghavan
(mkim, raghavan)@cacs.louisiana.edu
The Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504, USA

Abstract

There is considerable interest in bridging the gap between the terminology used in defining queries and the terminology used in representing documents. Some approaches use rules to capture user query concepts. The rules are usually weighted and expressed by AND/OR logical connectives. One difficulty with these approaches is that the resulting performance is quite sensitive to the weight assignments. We develop a neural network model in which the rule weights can be adjusted by users' relevance feedback. Experiments are performed on a small document collection and the adjusted rules show excellent performance in terms of recall-precision values.

1. Introduction

Many intelligent retrieval approaches [3, 7, 8] have tried to bridge the terminological gap that exists between the way in which users specify their information needs and the way in which queries are expressed. One proposed approach involves using production rules to capture user query concepts. The main ideas underlying such an approach were introduced in a system called Rule Based Information Retrieval by Computer (RUBRIC) [1, 4, 7]. In RUBRIC, a set of related production rules is represented as an AND/OR tree, called a rule base tree. RUBRIC allows the definition of detailed queries starting at a conceptual level. The retrieval output is determined by fuzzy evaluation of the AND/OR tree. However, since the resulting performance is quite sensitive to how weights are assigned to the rules, we need to find the proper weight values. For this purpose, we develop a neural network model in which the weights for rules can be adjusted by users' relevance feedback.

In RUBRIC, rules can be weighted and expressed by AND/OR logical connectives. This implies that the conventional Boolean retrieval technology is not sufficient to encode RUBRIC rules. There are some extended Boolean retrieval approaches [2, 10, 11] that are suitable to represent rules used in RUBRIC. Among the extended Boolean approaches, we adopt the p-norm based extended Boolean retrieval approach [10], since this approaches is not only theoretically well established, but also suitable to be mapped into a well known neural network model, called the multi-layered perceptron [6]. Our approach is different from the previous neural network approaches for information retrieval [12, 14] in at least the following two aspects. In the previous approaches, they were mainly concerned with term associations. In our approach, we handle relations between concepts and Boolean expressions in which weighted terms are involved. Secondly, in the most of the previous approaches, they introduced their own network model in order to map the retrieval problems into the neural network domain. In our approach, we use an already proven neural network model in terms of its performance.

* This research was supported in part by the U.S. Department of Energy, Grant No. DE-FG02-97ER1220.
+ On leave from the Department of Computer Engineering in Ajou University, Korea.

2. Review of RUBRIC

In RUBRIC, concepts of interest are formulated using a top-down refinement strategy. In this strategy, the first step is to express a given request as a single concept. The next step is to refine the initial concept by decomposing it into component parts that are related through either the AND or OR logical connective. Individual components may take the form of a new concept defined at a different abstraction level, a text expression, or a single index term. In each case, a weight value is assigned to the individual concept-component pairs that are formed during the decomposition process. The assigned weight value represents the user's belief in the degree to which a given component characterizes the related concept.

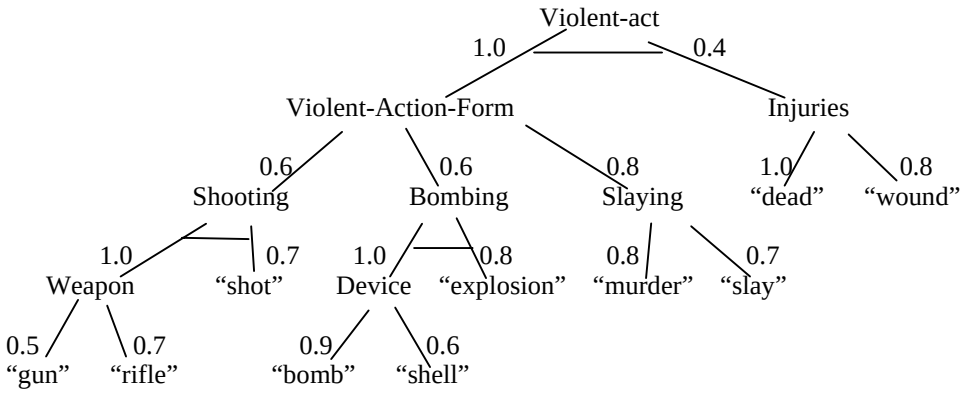

Fig 1. Rule base tree for concept Violent-act

Fig 1 shows a rule base tree for the concept **Violent-act** where the leaf nodes are index terms and are enclosed by double quotations, the internal nodes are concepts and the weights are displayed along the edges connecting concepts and components. The concept **Violent-act** is first being decomposed into two component concepts, **Violent-Action-Form** and **Injuries**, which are related to **Violent-act** with AND connective, denoted by drawing a line between its branches. If there is no line connecting the branches, the relationship is OR.

The relevancy evaluation (RSV: Retrieval Status Value) of a document to the query concept **Violent-act** could be processed by a bottom-up strategy. For example, if the document contains the words "gun," "shot," "bomb," "slay," and "dead" and no other words in the example rule base are referred to, then the index term nodes "gun," "shot," "bomb," "slay," and "dead" will receive weights of 1.0 and all other index term nodes will receive weights of 0. For example, consider the concept **Weapon** that is composed of two components "gun" and "rifle." The weight for "gun" is 1.0 and "rifle" is 0. Since the operator for **Weapon** is OR, the relevancy to concept **Weapon** is assigned to be the max value of the product of the weights of its components and the corresponding weights connecting its components. In this particular case, the result is Max (1.0*0.5, 0*0.7) = 0.5. The weight for concept **Shooting** could be calculated in the same way. Since the operator for **Shooting** is AND, we use the min function instead of the max function. The result is Min(0.5*1.0, 0.7*1.0) = 0.5. In a similar way, finally we can get the relevancy (RSV) of the document to the concept **Violent-act**, which comes out to be 0.3.

3. Review of P-Norm Based Extended Boolean Model

Salton *et al.*[10] proposed an extended Boolean model in order to overcome disadvantages in the conventional Boolean retrieval model [5]. Specifically, they introduce AND/OR logical connectives with weighted terms as follows. Consider a set of terms T_1, T_2, …., T_n and let a_i denote the weight of term T_i in a document $D = (a_1, a_2, \ldots, a_n)$ where $1 \le i \le n$ and $0 \le a_i \le 1$. Suppose that an or-query is given as $Q_{OR(p)} = OR^p(q_1, q_2, \ldots, q_n)$ where q_i indicates the weight of

query term T_i, $0 \le q_i \le 1$, and $1 \le p \le \infty$. Similarly, suppose that $Q_{AND(p)}$ is given as $Q_{AND(p)} = AND^p (q_1, q_2, \ldots, q_n)$. The similarities between a given document $D = (a_1, a_2, \ldots, a_n)$ and $Q_{OR(p)}$, $Q_{AND(p)}$ are defined as

$$sim(D, Q_{OR(p)}) = \left[\frac{q_1^p a_1^p + q_2^p a_2^p + ... + q_n^p a_n^p}{q_1^p + q_2^p + ... + q_n^p} \right]^{1/p} \quad (1)$$

$$sim(D, Q_{AND(p)}) = 1 - \left[\frac{q_1^p (1-a_1)^p + q_2^p (1-a_2)^p + ... + q_n^p (1-a_n)^p}{q_1^p + q_2^p + ... + q_n^p} \right]^{1/p} \quad (2)$$

In this extended Boolean model, when p = 1, the distinction between the AND and OR connectives disappears. In this case, the similarity between queries and documents can be computed by the inner product between their weights. This means that a simple vector space model [9] is obtained when p = 1. When $p = \infty$ and all query terms are equal to 1, we can obtain $sim(D, Q_{OR(\infty)}) = \max(a_1, a_2, \ldots, a_n)$ and $sim(D, Q_{AND(\infty)}) = \min(a_1, a_2, \ldots, a_n)$. By varying the value of p between 1 and ∞, it is possible to obtain a system intermediate between a pure vector model (p = 1) and a conventional Boolean retrieval system ($p = \infty$).[1]

4. Neural Network Model for Rule Base Tree

In this section, we propose a scheme to map a rule base tree into a multi-layered perceptron, using the p-norm based AND and OR connectives. The following two subsections describe how to construct the corresponding multi-layered perceptron and suggest its activation function and learning rules.

4.1 Neural Network Structure

Consider the rule base tree in Fig 1. Assuming the input term weights are given to the leaf nodes, we can compute the RSV for the documents that are indexed by the input terms. Actually, the computed RSV is nothing but the similarity between the query and the document where the query is represented in the rule base tree and the documents is indexed by the input terms. More simply, consider a unit tree in Fig 2.

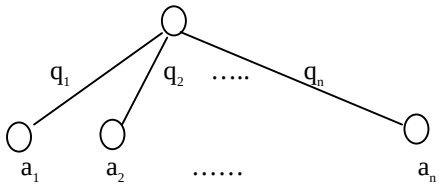

Fig 2. A unit rule base tree

Since the output node is an OR node, we can compute RSV = $\max(q_1a_1, q_2a_2, \ldots, q_na_n)$. If the output node is an AND node, the RSV will be $\min(q_1a_1, q_2a_2, \ldots, q_na_n)$.

Now, we transform a unit rule base tree into a unit neural network. The transformed unit neural network has the same topology of the given unit rule base tree. For example, the unit rule base tree in Fig 2 is transformed into the form in Fig 3. However, for the evaluation of RSV, we will use the right-hand sides in equation (1) and (2) instead of max and min functions, respectively. Therefore, net input value h_i for the node i can be defined in equation (3) and (4).

[1] For more information, see [10].

The input values are raised to the power p. The weight w_i for $1 \le i \le n_i$ is computed as in equation (5)

$$\text{If the node } i \text{ is an } OR \text{ node,} \quad h_i = \sum_j w_{ij} a_j^p \tag{3}$$

$$\text{If the node } i \text{ is an } AND \text{ node,} \quad h_i = \sum_j w_{ij} (1 - a_j)^p, \tag{4}$$

where w_{ij} is the weight from the node j to the node i and,

if node j is a hidden node then a_j is the activation of node j,

otherwise (i.e., j is an input node) a_j is the input value.

$$w_i = \frac{q_i^p}{q_1^p + q_2^p + ... + q_n^p} \qquad \text{for } 1 \le i \le n \tag{5}$$

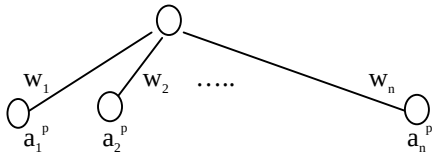

Fig 3. A unit neural network

Suppose that we transform the rule base tree in Fig 1 to a multi-layered neural network. Its structure and weights are as shown in Fig 4, where p = 5. To compute the weights in Fig 4, for example the weights between the output layer and the hidden layer 3, we use equation (5) and get the weights 0.99 ($1.0^5/(1.0^5 + 0.4^5) \cong 0.99$) and 0.01 ($0.4^5/(1.0^5 + 0.4^5) \cong 0.01$).

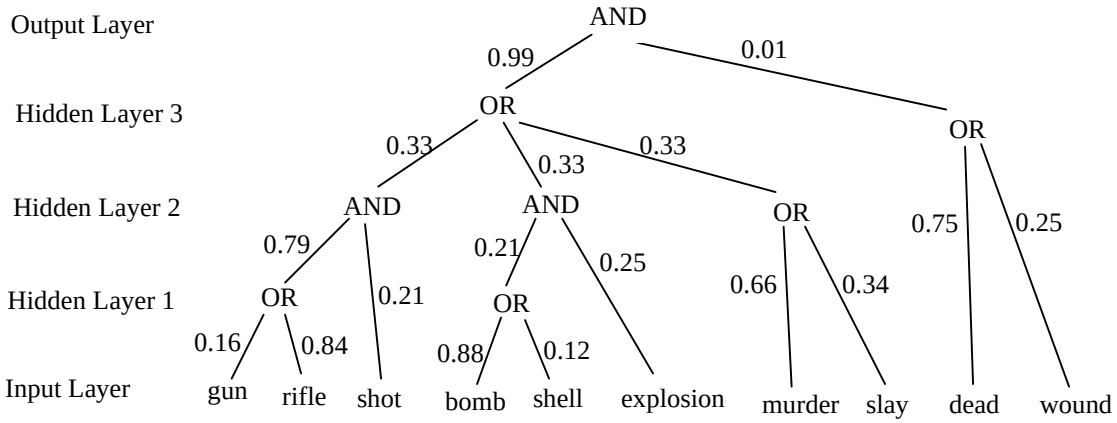

Fig 4. Neural network representation of the rule base tree in Fig 1.

Now, we can consider the activation functions for OR and AND nodes from equation (1) and (2), respectively, as follows.

$$a_i = F_{OR}(h_i) = h_i^{1/p} \qquad a_i = F_{AND}(h_i) = 1 - h_i^{1/p} \tag{6}$$

However, since these functions are not appropriate to be used as activation functions for the purpose of learning process, for reasons that will be explained later, we introduce approximation functions, in the next subsection, that can be effectively used during the learning process.

4.2 Activation Function and Learning Rule

From a mathematical point of view, the functions in equation (6), which have 1/p in the exponent, are exactly the correct choices to be used as the activation functions for the proposed neural network. However, since the derivative of such functions become extremely large near zero, it is difficult to make the learning process, to be described later, result in convergence. Therefore, we use an approximation of the 1/p powered functions that can easily get the convergence in the learning process. The adopted function is a variation of the sigmoid, which we call 2p-sigmoid function, defined as.

$$\hat{F}(x) = \frac{1}{1+e^{-2p(x-\theta)}}, \quad \text{where } \theta \text{ is 0.5.}$$

We use $\hat{F}(x)$[2] above, instead of $x^{1/p}$, to approximate the activation functions for OR nodes and AND nodes, respectively, as follows.

$$a_i = \hat{F}_{OR}(h_i) = \frac{1}{1+e^{-2p(h_i-\theta)}} \tag{7}$$

$$a_i = \hat{F}_{AND}(h_i) = 1 - \frac{1}{1+e^{-2p(h_i-\theta)}} \tag{8}$$

For the learning process, we use the back-propagation learning rule [6]. Its derivation is fairly straightforward. The error measure E is defined as the total quadratic function at the output nodes:

$$E = \frac{1}{2}\sum (d_i - a_i)^2,$$

where d_i and a_i are the desired output and the actual output for the node i, respectively. Then, By the standard back-propagation formulas [6], we can write Δw_{ij}:

$$\Delta w_{ij} = \gamma \delta_i a_j^p \quad \text{for } OR \text{ node} \quad \text{and} \quad \Delta w_{ij} = \gamma \delta_i (1-a_j)^p \quad \text{for } AND \text{ node } i$$

where $\gamma > 0$ is the learning rate and δ_i is the error signal given as follows. If the node i is an output node, δ_i is given by

$$\delta_i = (d_i - a_i)\hat{F}'(h_i)$$

When $\hat{F}_{OR}$ and $\hat{F}_{AND}$ are given as in equation (7) and (8), respectively, the derivatives are equal to

$$\hat{F}'_{OR}(h_i) = \frac{2p}{(1+e^{-2p(h_i-\theta)})^2} e^{-2p(h_i-\theta)} = 2pa_i(1-a_i)$$

[2] That F and $\hat{F}$ agree for many values of x can be verified by a simple tabulation.

$$\hat{F}'_{AND}(h_i) = \frac{-2p}{(1+e^{-2p(h_i-\theta)})^2} e^{-2p(h_i-\theta)} = 2pa_i(a_i - 1)$$

If the node i is a hidden node, the error signal is determined recursively in terms of error signals of the nodes to which it directly connects and the weights of those connections.[3] Therefore, the error signal is given by

$$\delta_i = \hat{F}'(h_i)\sum_k \delta_k w_{ki}, \text{ where k is over all nodes in the layers above node i.}$$

5. Experimental Tests

The main purpose of our experiment is to determine whether our proposed neural network would be adaptable to a given set of training samples, and whether the recall-precision is improved with the adapted weights. For this purpose, we use a collection of 196 documents retrieved by the Google search engine [15] with the index terms, that are used to define the concept **Violent-act** as represented in Fig 1, submitted as an OR query. In this collection, the relevance feedback is given by having a graduate student look through each document and decide whether or not it is relevant to the query concept. From the collection, we randomly chose half of the documents (98 documents) to as the training sample set. The remaining half of the documents (98 documents) was used as the testing sample set. We have conducted several tests with different rule base trees. The results usually show similar performance. In this paper, we report a test with the same weights in Fig 1 and value 0.7 as the desired output for relevant documents and value 0.4 for irrelevant documents. With the training sample set, we repeatedly perform the back-propagation while the error *E* is decreasing. Experimentally, the number of epochs usually does not exceed 20. However, we find that the learning process is not stable in some cases, specifically, in which the training sample set is not good and/or the initial weight values are too small.

From this learning process, we can get the adjusted weights. Since we use the calculated weights for the neural network as in equation (5), we need to convert them back to the weights used in rule base trees. However, since there is no analytic way for the conversion, we compute the weight under the following assumption. Let $q_1, q_2, \ldots, q_n$ be the weights between input nodes and output node in a rule base tree. The translated weight w_i for the corresponding neural network is given by equation (5). Let the adjusted weight of w_i be w_i', and the weight to be adjusted from q_i be r_i. We assume that

$$q_1^p + q_2^p + ... + q_n^p = r_1^p + r_2^p + ... + r_n^p = C_p, \quad \text{where } C_p \text{ is some constant.} \qquad (11)$$

Then, we can write as

$$w_{ij} = \frac{q_j^p}{C_p} \quad \text{and} \quad w'_{ij} = \frac{r_j^p}{C_p}.$$

$$\text{Therefore, } r_j^p = q_j^p \frac{w'_{ij}}{w_{ij}}.$$

After 8 epochs with p = 5, we can get the adjusted weights as in Fig 5. With the testing set (98 documents), we compute the recall-precision values with the original weights and the adjusted

[3] For detailed information, see [6].

weights, respectively. We also compute the recall-precision values using the original RUBRIC approach. The result is shown in Table 2.

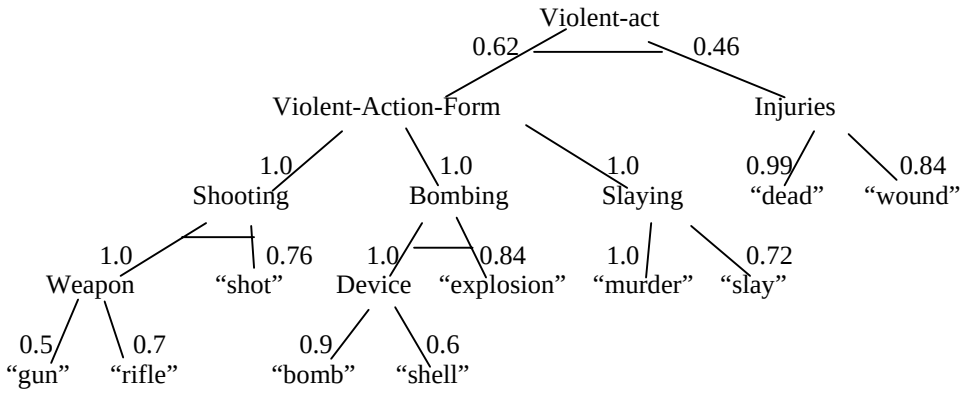

Fig 5. Rule base tree with adjusted weights in Test 1.

Table 2. Recall-precision in Test 1

Recall	Precision			
	Before learning		After learning	
	RUBRIC	Neural Model	RUBRIC	Neural Model
0.1	0.7143	0.5000	0.6250	1.0000
0.2	0.6667	0.6667	0.6667	0.8333
0.3	0.5357	0.6818	0.6522	0.8333
0.4	0.5714	0.6667	0.6897	0.6061
0.5	0.5952	0.7143	0.7143	0.5682
0.6	0.5769	0.7317	0.7143	0.5769
0.7	0.6071	0.6538	0.5667	0.5965
0.8	0.5278	0.6333	0.5352	0.6230
0.9	0.5122	0.5250	0.5316	0.6000
1.0	0.4894	0.4792	0.4894	0.4894
Average improvement over RUBRIC			7.4%	16.1%

As shown in Table 2, the precision values are much better for our neural network model with the adjusted weights.

6. Conclusions

From the experiments, we can reach the following conclusions.

(1) Multi-layered perceptron and back-propagation rule can be effectively used for adjusting the rule weights for defining concepts.
(2) The adjusted rule weights usually show excellent recall-precision values in our neural network model, but this is not the case in the original RUBRIC approach.

Our research can be extended in the following directions. We need more experimental tests and need to find a way to adjust the rules that work universally. A way to perform further experiments might be to change p value the in the p-norm based extended Boolean approach and/or to find more relevant activation functions for the neural work. A more challenging

problem is to develop mechanisms that allow to learn more appropriate structures of rules, not just proper weights of the rules.

References

1. Alsaffar, A. H., Deogun, J. S., Raghavan, V. V., and Sever, H. Concept-based retrieval with minimal term sets. In Z. W. Ras and A. Skowon, editors, Foundations of Intelligent Systems: Eleventh Int'l Symposium, ISMIS'99 proceedings, pp. 114-122, Springer, Warsaw, Poland, Jun, 1999.
2. Bookstein, A. Fussy requests: An approach to weighted Boolean searches, *J. ASIS*, Vol 31, No. 4, July, 1980, pp. 275-279
3. Croft, W. B. Approaches to intelligent information retrieval. Information Processing and Management, 1987, Vol. 23, No. 4, pp. 249-254.
4. Fenghua Lu, Thomas Johnsten, Vijay Raghavan and Dennis Traylor, Enhancing Internet Search Engines to Achieve Concept-based Retrieval, Proceeding of Inforum'99, Oakridge, USA.
5. Lancaster, F. W. Information retrieval systems: characteristics, testing and evaluation, 2nd Ed., John Wiely and Sons, New York, 1979.
6. Lippmann, R. P. An introduction to computing with neural nets. *IEEE ASSP Magazine*, Vol. 3, No. 4, pp. 4-22.
7. McCune, B. P., Tong, R. M., Dean, J. S., and Shapiro, D. G. RUBRIC: A System for Rule-Based Information Retrieval, *IEEE Transaction on Software Engineering*, Vol. SE-11, No. 9, September 1985.
8. Resnik, P. Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 448-453, 1995.
9. Salton, G. and McGill, M. J. *Introduction to Modern Information Retrieval*. McGraw Hill, New York, 1983.
10. Salton, G., Fox, E. A., and Wu, H. Extnded Boolean Information Retrieval, Vol. 36, No. 11, December 1983, Communication of the ACM, pp. 1022-1036.
11. Waller, W. G. and Kraft, D. H. A mathematical model for a weighted Boolean retrieval system. Information Processing and Management, Vol 15, No. 5, 1979, pp. 235-245.
12. Wilkinson, R. and Hingston, P. Using the cosine measure in a neural network for document retrieval. In Proceedings of ACM-SIGIR Conference, 1991, pp. 202-210.
13. Wong, S.K.M., Ziarko, W., Raghavan,V., and Wong, P. C. N. Extended Boolean Query Processing in the Generalized Vector Space Model, *Information Systems* Vol. 14, No. 1, pp. 47-63, 1989.
14. Wong, S.K.M. and Cai, Y.J. Computation of term associations by a neural network. In Proceedings of ACM-SIGIR Conference, 1993, pp. 107-115.
15. "Google Search Engine," http://www.google.com.