Specialized Neural Network and Fuzzy Logic Algorithm in Comparison for Word Frequency Prediction in Document Filtering [1]

Péter Baranyi, Dept. of Automation, T. U. of Budapest, Hungary, baranyi@elektro.get.bme.hu
Petra Aradi, Department of Systems and Control Engineering, T.U. of Budapest, Hungary
László T. Kóczy, Dept. of Telecommunication and Telematics, T. U. of Budapest, Hungary
Tamás D. Gedeon, School of Computer Science and Eng., U. of New South Wales, Australia

Abstract

For very large document collections or high volume streams of documents, finding relevant documents is a major information filtering problem. A major aid to information retrieval systems produces a word frequency measure estimated from some important parts of the document using neural network approaches. In this paper a fuzzy logic technique and as its simplified case a neural network algorithm will be proposed for this task. The comparison of these two and an alternative neural network algoritm will also be discussed.

1. Introduction

An information system allows users to efficiently retrieve documents from a very large collection, that are relevant to their current interests [7]. The information retrieval system has to determine the relevant documents by using the matching of key words or phrases of user interest specified in the query, that are fragments of natural language texts, and the words and phrases used in documents of the collection. The main difficulty is that an effective search for matching has to consider the whole topic determined by the queried words and their synonyms, or else relevant documents are lost. If e.g. only synonyms are used in the text of certain documents which are very relevant, these documents might be completely left out of consideration. In order to alleviate this problem, many research works have emerged on the field of creating and maintaining information filters [10,11,12,13 and 14], specifying categories, building synonym lists, and so on [8]. In these systems it is inevitable to deal with the problem of automatic indexing [1,2,5,6]. The aim of indexing text items is to (implicitly) summarize their content [9]. Once important keywords are found and their occurrence frequencies are known or estimated, it is possible to build up co-occurrence maps and hierarchical co-occurrence relations [2,3,16]

The proposed automatic indexing algorithms are based on the combination of some significant measures such as frequency-keyword approach, title-keyword, location-method and cue-method [2,3]. One of the main ideas to determine the most appropriate way of combining the indexing parameters is based on neural network approaches that can learn a composition function of significance measures. We train a neural network to estimate the word frequency measure component of a retrieval index from a relatively small proportion of the document texts (e.g. the first and last 20%), so, all of these measures can be calculated locally and do not require global document collection information as it is required for the real frequency-keyword calculations. This may provide considerable gains in the future in highly parallel implementation [15]. Consequently, the efficiency of information retrieval significantly depends on the effectiveness of the frequency-keyword measure, hence, the applied algorithms [17].

In this paper a fuzzy logic algorithm (FLA) will be proposed with examples for the estimation of the word frequency measure component in comparison to neural network algorithms. It will be shown that this algorithm also can be introduced as a neural network, where the neurons can include different piece-wise linear transfer functions. A simple neural network (SNN) as a special case of the proposed FLA will also be discussed with examples. These algorithms are optimized considering the main difficulties of these kinds of applications, namely, the calculation time complexity.

The advantage of applying neural networks or fuzzy logic technique lies in their ability to imitate and implement the actions of expert operator(s) without the need of accurate mathematical models. The drawback, however, is that there is no standardized framework

[1] Supported by the Australian Research Council, Grant No. A49600961

0-7803-4863-X/98 $10.00©1998 IEEE

regarding the design, optimality, reducibility, and definition of the concept. These algorithms, either generated by expert operators, or by some learning or identification schemes, may contain redundant, weakly-contributing, or unnecessary components. Moreover, to achieve a good approximation, some approaches may overly estimate the number of neural network hidden units and layers or the number of fuzzy rules, thereby resulting in a large neural network or fuzzy rule base causing problems in computational time complexity. Applying learning algorithms to a large number of neurons or having a large number of rules in a fuzzy rule base needs considerable calculation time even in small applications

As mentioned above, the problem is that the collection of documents, from which the selected ones have to be retrieved might be extremely large. Thus, two important aims have to be taken into consideration to generate a neural network or fuzzy logic representation. One is to achieve a sufficient estimation of the frequency of considered words including their co-occurrence relations. The other is to use simplified functions in the neurons and to reduce the number of layers and neurons or fuzzy rules as much as possible to reduce the required computational effort.

We will first describe a standard neural network approach for estimating the word frequency [1,2,5,6]. The algorithm proposed in this paper has reduced computational time complexity. The SNN has less neurons, and it has simplified functions in the units rather than the usual sigmoid function requiring an exponential operation. The advantages and disadvantages of the FLA and SNN will be discussed below.

Consequently, using the proposed methods the learning time is considerably reduced however the obtained results are significantly improved.

2. Standard neural network approach

Let us take the last improved network from [2]. The selection of the words and training parameters were as follows: $n_w = 75$ words were used to index the collection of 350 documents. The chosen neural network had three inputs ($n_p = 3$) for each 75 words. These were for the title-keywords, location- and cue-frequencies. Thus, the total number of input was $I = n_w n_p = 225$. The input value is $x_i \in [0,1]$, where $i = 1..I$. The output values $y_k \in [0,1]$ $(k = 1,\ldots,n_w)$ were the estimated frequency-keyword measures for each word. The number of output neurons was $O = 75$. The number of hidden neurons was $H = 4$. The transfer function included in the neurons was:

$$f(a) = \frac{1}{1+e^{-a}}$$

The essence of the experiment was to use these measures as training inputs to the network, and then to attempt the prediction of the frequency-keyword measures.

The network input vector is $\underline{\mathbf{x}} = [x_i]\,(i = 1..s..I$ where $s = (i_w - 1)n_w + i_p$, and $i_w = 1,\ldots,n_w$, $i_p = 1,\ldots,n_p,)$. The output value of input neuron N_i^I is $y_i^I = x_i$, of hidden neuron N_j^H is y_j^H $(j = 1...H)$ and of output neuron N_k^O is $y_k\,(k = 1...O)$. The connection weight matrix between neurons of the input and hidden layers is $\underline{\underline{\mathbf{W}}}^I = \left[w_{j,i}^I\right]$, between the hidden and output layers: $\underline{\underline{\mathbf{W}}}^H = \left[w_{k,j}^H\right]$. The hidden and output layers are biased, that means there is an additional neuron in both the input and hidden layers with a constant output of 1. The bias weight vector for the hidden layer is $\underline{\mathbf{w}}^{I,bias} = \left[w_j^{I,bias}\right]$, for the output layer $\underline{\mathbf{w}}^{H,bias} = \left[w_k^{H,bias}\right]$.The input values of neuron N_j^H in the hidden layer are $x_{j,i}^H = w_{j,i}^I y_i^I$, and from the bias neuron $x_{j,0}^H = w_j^{I,bias} \cdot 1$. Similarly, the input values of neuron N_k^O in the output layer are $x_{k,j}^O = w_{k,j}^H y_j^H$, and from the bias neuron $x_{k,0}^O = w_k^{H,bias} \cdot 1$.

The steps of forward propagation were as follows. Let $\underline{\tilde{\mathbf{y}}}$ be the output calculated by the network as:

$$1.\ \underline{\mathbf{y}}'^H = \left[\underline{\underline{\mathbf{W}}}^I\ \underline{\mathbf{w}}^{I,bias}\right]\begin{bmatrix}\underline{\mathbf{y}}^I\\ 1\end{bmatrix} \qquad (2)$$

where $\underline{\mathbf{y}}'^H$, $\underline{\underline{\mathbf{W}}}^I$, $\underline{\mathbf{w}}^{I,bias}$ and $\underline{\mathbf{y}}^I$ contain elements $y_j'^H$, $w_{j,i}^I$, $w_j^{I,bias}$ and y_i^I, respectively.

$$2.\ y_j^H = f\left(y_j'^H\right) = \frac{1}{1+e^{-y_j'^H}},\quad 3.\ \underline{\mathbf{y}}' = \left[\underline{\underline{\mathbf{W}}}^H\ \underline{\mathbf{w}}^{H,bias}\right]\begin{bmatrix}\underline{\mathbf{y}}^H\\ 1\end{bmatrix}$$

where $\underline{\mathbf{y}}'^O$, $\underline{\underline{\mathbf{W}}}^H$, $\underline{\mathbf{w}}^{H,bias}$ and $\underline{\mathbf{y}}^H$ contain elements $y_k'^O$, $w_{k,j}^H$, $w_k^{H,bias}$ and y_j^H, respectively.

$$4.\ \tilde{y}_k = f\left(y_k'\right) = \frac{1}{1+e^{-y_k'}},\ 5.\ \text{The error is } \underline{\delta}^O = \underline{\mathbf{y}} - \underline{\tilde{\mathbf{y}}}$$

Thus, based on error back propagation

$$6.\ \delta_k'^O = \tilde{y}_k\left(1-\tilde{y}_k\right)\delta_k^O,\qquad 7.\ \underline{\delta}^H = \underline{\underline{\mathbf{W}}}^{H\mathrm{T}}\,\underline{\delta}'^O$$

$$8.\ \delta_j'^H = y_j^H\left(1-y_j^H\right)\delta_j^H,\quad 9.\ \Delta\left[\underline{\underline{\mathbf{W}}}^H\ \underline{\mathbf{w}}^{H,bias}\right] = \eta\underline{\delta}'^O\,\underline{\mathbf{y}}^{H\mathrm{T}}$$

10. $\Delta\left[\underline{\underline{\mathbf{W}}}^{I}\ \underline{\mathbf{w}}^{I,bias}\right] = \eta\underline{\delta}'^{H}\mathbf{y}^{I\mathrm{T}}$

where $0<\eta<1$ is the learning parameter. After training the network has formed an internal representation of the relative importance of the different significance measures.

3. The simplified neural network (SNN)

In order to reduce the calculation time as opposed to the standard approach described in section 2 and applied in our former approach in [5,6], the transfer function of the neurons, which is an exponential function requiring relatively much computation, is omitted. From the nature of the problem we concluded that bias weights may not be necessary, as zero occurences of any particular word have no effect on the significance of other words. If a document included most of the possible words, we could clearly say that any exceptions were significant. Since real documents contain few words, the absence of any particular word compared to the number of all possible absent words provides little real information. Hence we can ignore the bias weights as the neural network hyperplanes necessarily pass through or near the origin. As a matter of fact, in this case the hidden layer does not have any role and the output is calculated without steps 3 and 5 (1) and with bias inputs set to 0:

$$\underline{\tilde{\mathbf{y}}} = \underline{\underline{\mathbf{W}}}^{H}\underline{\underline{\mathbf{W}}}^{I}\underline{\mathbf{x}} = \underline{\underline{\mathbf{W}}}\underline{\mathbf{x}} \qquad (3)$$

However, the hidden layer can be used for computational complexity reduction. The size of $\underline{\underline{\mathbf{W}}}^{I}$ is $I\times H$, the size of $\underline{\underline{\mathbf{W}}}^{H}$ is $H\times O$. If only $\underline{\underline{\mathbf{W}}}$ is used the total number of connections is $I\cdot O$. Thus, if $H<(I\cdot O)/(I+O)$ then the connection complexity is reduced as compared to the case when only $\underline{\underline{\mathbf{W}}}$ is used.

Singular value decomposition (SVD) is applied to choose the optimal H in the sense of connection complexity reduction. Applying SVD on $\underline{\underline{\mathbf{W}}}$ results in $\underline{\underline{\mathbf{W}}} = \underline{\underline{\mathbf{U}}}\underline{\underline{\mathbf{D}}}\underline{\underline{\mathbf{V}}}^{\mathrm{T}}$ where $\underline{\underline{\mathbf{U}}}$ and $\underline{\underline{\mathbf{V}}}$ are orthogonal matrices with size $n_u\times n_u$ and $n_v\times n_v$ respectively. The $n_u\times n_v$ $\underline{\underline{\mathbf{D}}}$ matrix contains the singular values of $\underline{\underline{\mathbf{W}}}$ in decreasing order. The maximum number of non-zero singular values is $n_{SVD}=\min(n_u,n_v)$. The singular values indicate the importance of the corresponding columns in matrices $\underline{\underline{\mathbf{U}}}$ and $\underline{\underline{\mathbf{V}}}$. To achieve a simpler connection, the singular values equal or near to zero should be discarded resulting in an approximation of $\underline{\underline{\mathbf{W}}}$. Let n_r be the number of singular values decided to be retained. Let us partition the matrices $\underline{\underline{\mathbf{U}}}$, $\underline{\underline{\mathbf{V}}}$ and $\underline{\underline{\mathbf{D}}}$ as $\underline{\underline{\mathbf{U}}}=\left|\underline{\underline{\mathbf{U}}}_{r}\ \ \underline{\underline{\mathbf{U}}}_{d}\right|$, $\underline{\underline{\mathbf{V}}}=\left|\underline{\underline{\mathbf{V}}}_{r}\ \ \underline{\underline{\mathbf{V}}}_{d}\right|$ and $\underline{\underline{\mathbf{D}}}=\begin{vmatrix}\underline{\underline{\mathbf{D}}}_{r} & \underline{\underline{\mathbf{O}}}_{n_r\times(n_v-n_r)}\\ \underline{\underline{\mathbf{O}}}_{(n_u-n_r)\times n_r} & \underline{\underline{\mathbf{D}}}_{d}\end{vmatrix}$.

Here, $\underline{\underline{\mathbf{O}}}_{a\times b}$ is an $a\times b$ matrix of zeros, and $\underline{\underline{\mathbf{U}}}_{r}$, $\underline{\underline{\mathbf{U}}}_{d}$, $\underline{\underline{\mathbf{V}}}_{r}$ and $\underline{\underline{\mathbf{V}}}_{d}$ are $n_u\times n_r$, $n_u\times(n_u-n_r)$, $n_v\times n_r$ and $n_v\times(n_v-n_r)$ matrices, respectively. Matrix $\underline{\underline{\mathbf{D}}}_{r}$ is diagonal, and it contains the retained n_r singular values. $\underline{\underline{\mathbf{U}}}_{r}$ and $\underline{\underline{\mathbf{V}}}_{r}$ contain the corresponding columns of $\underline{\underline{\mathbf{U}}}$ and $\underline{\underline{\mathbf{V}}}$. Matrix $\underline{\underline{\mathbf{D}}}_{d}$ contains the remaining $(n_{SVD}-n_r)$ singular values to be discarded. An approximation of $\underline{\underline{\mathbf{W}}}$, $\underline{\underline{\tilde{\mathbf{W}}}}$ is obtained by $\underline{\underline{\tilde{\mathbf{W}}}}=\underline{\underline{\mathbf{U}}}_{r}\underline{\underline{\mathbf{D}}}_{r}\underline{\underline{\mathbf{V}}}_{r}^{\mathrm{T}}$. The approximation is exact if all non zero singular values are kept as $\underline{\underline{\mathbf{W}}}=\underline{\underline{\mathbf{U}}}_{r}\underline{\underline{\mathbf{D}}}_{r}\underline{\underline{\mathbf{V}}}_{r}^{\mathrm{T}}$. Consequently, if $n_r<(I\cdot O)/(I+O)$ then let $\underline{\underline{\mathbf{W}}}^{I}=\underline{\underline{\mathbf{U}}}_{r}$ and $\underline{\underline{\mathbf{W}}}^{H}=\underline{\underline{\mathbf{D}}}_{r}\underline{\underline{\mathbf{V}}}_{r}$ or obviously $\underline{\underline{\mathbf{W}}}^{I}=\underline{\underline{\mathbf{U}}}_{r}\underline{\underline{\mathbf{D}}}_{r}$ and $\underline{\underline{\mathbf{W}}}^{H}=\underline{\underline{\mathbf{V}}}_{r}$. If n_r is not less than $(I\cdot O)/(I+O)$, non-zero singular values can be discarded. In this case the error bound of SVD reduction have to be considered.

As it is mentioned above, if $\underline{\underline{\mathbf{D}}}_{d}$ contains non-zero elements the product of retained matrices is an approximation of $\underline{\underline{\mathbf{W}}}$. So, $\underline{\underline{\mathbf{W}}}-\underline{\underline{\tilde{\mathbf{W}}}}=\underline{\underline{\mathbf{U}}}_{d}\underline{\underline{\mathbf{D}}}_{d}\underline{\underline{\mathbf{V}}}_{d}^{\mathrm{T}}$. As the columns of $\underline{\underline{\mathbf{U}}}_{d}$ and $\underline{\underline{\mathbf{V}}}_{d}$ have the Euclidean norm of unity, the absolute values of their elements must be bounded by 1. Denote σ_i the i-th singular value of $\underline{\underline{\mathbf{D}}}$, and we obtain

$$\left|\underline{\underline{\mathbf{W}}}-\underline{\underline{\tilde{\mathbf{W}}}}\right| = \left|\underline{\underline{\mathbf{U}}}_{d}\right|\left|\underline{\underline{\mathbf{D}}}_{d}\right|\left|\underline{\underline{\mathbf{V}}}_{d}^{\mathrm{T}}\right| \le 1_{n_u\times(n_u-n_r)}\underline{\underline{\mathbf{D}}}_{d}1_{(n_v-n_r)\times n_v}$$

$$1_{n_u\times(n_u-n_r)}\underline{\underline{\mathbf{D}}}_{d}1_{(n_v-n_r)\times n_v} = \left(\sum_{i=n_r+1}^{n_{SVD}}\sigma_i\right)1_{n_u\times n_v} \qquad (3)$$

It is not necessary to take the absolute value of $\underline{\underline{\mathbf{D}}}_{d}$, as each singular value is positive. Equation (3) implies $\left|w_{i_l,i_{l-1}}-\tilde{w}_{i_l,i_{l-1}}\right|\le d=\sum_{i=n_r+1}^{n_{SVD}}\sigma_i$. Thus, $\left|y_i-\tilde{y}_i\right|\le\sum_{i=1}^{I}dx_i$, where I is the number of inputs.

The learning phase of the application results in a strongly non-singular matrix $\underline{\underline{\mathbf{W}}}$. Discarding only one singular value results in a large error. (In our case $(I\cdot O)/(I+O)=56.25$, that means at least 19 singular values have to be discarded for reduction.) This is quite obvious from the behavior of the training parameters.

Suppose that the network has one hidden layer, where the number of neurons is H.

Let us have a look at the matrices $\underline{\underline{\mathbf{Y}}}_T = \left[\underline{\mathbf{y}}_1 \dots \underline{\mathbf{y}}_{n_t}\right]$ and $\underline{\underline{\mathbf{Y}}}_{T2} = \left[\underline{\mathbf{y}}_1^H \dots \underline{\mathbf{y}}_{n_t}^H\right]$, where $\underline{\mathbf{y}}_t^H$ contains the output values of the hidden layer for the t-th training input. In case of ideal training we obtain $\underline{\underline{\mathbf{Y}}}_T = \underline{\underline{\mathbf{W}}}^H \underline{\underline{\mathbf{Y}}}_{T2}$, where $n_t >> O$ and $n_t >> H$. Suppose that $H < I$ and $H < O$. In this case the rank of $\underline{\underline{\mathbf{W}}}^H$ and $\underline{\underline{\mathbf{Y}}}_{T2}$ is maximum H. Thus, the rank of $\underline{\underline{\mathbf{W}}}^H \underline{\underline{\mathbf{Y}}}_{T2}$ is maximum H. As the training parameters are from randomly chosen sample documents, the rank of $\underline{\underline{\mathbf{Y}}}_T$ is almost certainly larger than H, but not larger than O. This matrix is most probably non-singular, that means H must be close to O. In our case $O = 75$, so, H must be close to 75. For the reduction H must be less than 57 (as calculated above), or else the use of a hidden layer increases the number of connections. Consequently, we do not use any hidden layer.

In the same way it can be seen that the number of neurons (4) in the hidden layers in the standard network approach is also not enough. It should have been closer to 75. The neurons in the former networks contained exponential functions, and with 75 neurons in the hidden layer instead of 4, this would have resulted in a considerable increase of calculation time.

Let us assume the algorithm of the proposed SNN.

1) $\underline{\tilde{\mathbf{y}}} = \underline{\underline{\mathbf{W}}}\underline{\mathbf{x}}$, 2) $\underline{\delta} = \underline{\mathbf{y}} - \underline{\tilde{\mathbf{y}}}$, 3) $\Delta\underline{\underline{\mathbf{W}}} = \eta\underline{\delta}\underline{\mathbf{x}}^{\mathrm{T}}$

It can be seen that SNN is much simplified comparing to (2). Negative word frequency measurements are not interpretable. As the new method may result in negative values the output must be bounded below.

4. Fuzzy logic approach (FLA)

The characterization of the input output is the same as in the SNN.

In order to save the computational effort we use triangular fuzzy sets as $\left\{A_{i,k=1} \ \dots \ A_{i,k=K_i}\right\}$, where K_i is the number of antecedent sets on input universe $X_i = [0,1]$ in such a way that $\mathrm{Core}\{A_{i,k}\} = a_{i,k}$, $\mathrm{support}\{A_{i,k}\} = [a_{i,k-1}, a_{i,k+1}]$, $\mathrm{support}\{A_{i,1}\} = [a_{i,1}, a_{i,2}]$, $\mathrm{support}\{A_{i,K_i}\} = [a_{i,K-1}, a_{i,K_i}]$.

Considering that input values are most frequently between 0 and 0.1 on each input universe let $a_{i,1} = a_1 = 0$, $a_{i,2} = a_2 = 0.05$, $a_{i,3} = a_3 = 0.1$, $a_{i,4} = a_4 = 1$ and $K_i = K = 4$. The consequent fuzzy sets $B_{o,m}$ ($0 = 1..n_w$ and $m = 1..M$, where M is the number of consequent sets on each output universe Y_o) are singleton sets as: $\mu_{B_{o,m}}(y_o) = \delta(y_{o,m})$; $y_o \in Y_o$. For the observation A^*_i we use singleton set as $\mathrm{core}\{A^*_i\} = x_i$ where x_i is the input value on X_i. The number of rules obtained by the all combination of antecedent sets is $\prod_i K_i = K^I = 4^{225}$ that can not be applied. In order to reduce the rule base we consider only the rules as following:

If $A_{i,k}$ then $B_{o,m} \Rightarrow \delta(y_{o,m})$

where $m = (i-1)K + k$, so $M = I \cdot K$. Thus the number of rules is 67500. We use fuzzy inference based upon product-sum-gravity [4]. Thus the output values is calculated as: $y_o = \dfrac{\sum_{i,k} \mu_{A_{i,k}}(x_i) y_{o,m}}{\sum_{i,k} \mu_{A_{i,k}}(x_i)}$; (4)

Membership degrees of the antecedent sets at any value within the universe sum to 1. Thus $\sum_{j,k} \mu_{A_{j,k}}(x_i) = I$ that means this rule base a piece-wise linear approximation as from (4) $y_o = \sum_{i,k} \mu_{A_{i,k}}(x_i) \dfrac{y_{o,m}}{I}$. To train the fuzzy rule base the same algorithm can be used as for SNN. The learning method does not tune all set, but only the position of the consequent sets. Let $\underline{\mathbf{m}}$ is a vector that contains the membership degrees as: $\underline{\mathbf{m}} = \left[m_1 \dots m_m \dots m_M\right]$, where $m_m = \mu_{A_{i,k}}(x_i)$ and $m = (i-1)K + k$. Matrix $\underline{\underline{\mathbf{B}}} = [y_{o,m}]$ contains the core of $B_{o,m}$. The steps of the algorithm are:

1) $\underline{\tilde{\mathbf{y}}} = \underline{\underline{\mathbf{B}}}\underline{\mathbf{m}}$, 2) $\underline{\delta} = \underline{\mathbf{y}} - \underline{\tilde{\mathbf{y}}}$, 3) $\Delta\underline{\underline{\mathbf{B}}} = \eta\underline{\delta}\underline{\mathbf{m}}^{\mathrm{T}}$,

It is much simpler than (2). In our application if an input parameter has zero value in this case it has zero contribution to the output value, that implies, that the position of consequent of rules If $A_{i,1}$ then $B_{o,m=(i-1)K+1}$ is zero, namely, values $y_{o,(i-1)K+1}$ are not tuned.

5. Comparison of SNN and FLA

The number of trained parameters in SNN is $I \cdot O$. In FLA it is $I \cdot O \cdot K$. Thus FLA needs more than K times more calculation time, however, much better result can be obtained. Equation (4) can be written as:

$$y_o = \sum_{i,k} \mu_{A_{i,k}}(x_i) \frac{y_{o,m}}{I} = \frac{1}{I} \sum_i f_{i,o}(x_i)$$

$$f_{i,o}(x_i) = \sum_k \mu_{A_{i,k}}(x_i) y_{o,m} = \frac{\sum_k \mu_{A_{i,k}}(x_i) y_{o,m}}{\sum_k \mu_{A_{i,k}}(x_i)}$$

where $\sum_k \mu_{A_{i,k}}(x_i) = 1$ as mentioned above. Thus $f_{i,o}(x_i)$ can be considered as an explicit function of rule base where the number of rules is K (Fig. 1). This function is the contribution function of i-th input universe to o-th output.

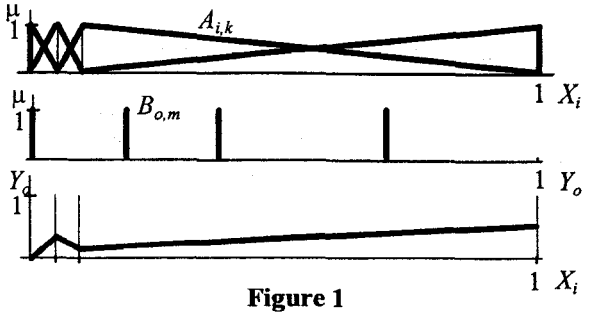

Figure 1

FLA can be represented as a neural network, where the units have different type function linearly approximated. SNN uses linear contribution function:

$y_o = \sum_i f_{i,o}(x_i)$, where $f_{i,o}(x_i) = w_{o,i}x_i$

It can be said consequently, that proper type contribution function can be trained for each input - output in FLA, it can be considered as the sum of proposed kind neural network as: $y_o = \frac{1}{I}\sum_i f_{i,o}(x_i) = \sum_i f'_{i,o}(x_i)$;where

$$f'_{i,o}(a) = \begin{cases} f_{1,i,o}(a) = w_{1,i.o}a & \text{if} a_{i,1} \le a < a_{i,2} \\ f_{2,i,o}(a) = w_{2,i.o}a + w^b_{2,i,o} & \text{if} a_{i,2} \le a < a_{i,3} \\ \vdots & \vdots \\ f_{K-1,i,o}(a) = w_{K-1,i.o}a + w^b_{K-1,i,o} & \text{if} a_{i,K-1} \le a < a_{i,K} \end{cases}$$

otherwise $f_{1,i,o} = 0$
otherwise $f_{2,i,o} = 0$
$\vdots$ $\vdots$
otherwise $f_{K-1,i,o} = 0$

Thus $y_o = \sum_{z=1}^{K-1}\sum_i f_{z,i,o}(x_i)$, which is the sum of K-1 neural networks, where the input neurons have bias input value $w^b_{z,i,o}$ except in the first network.

From this viewpoint the SNN is a special case of FLA. Namely, SNN is a fuzzy algorithm where the number of antecedent sets on each input universe is K=2 and the support of them is $[a_{i,1} = 0, a_{i,2} = 1]$. Consequently, FLA is more powerful description than SNN. The disadvantage is, that using FLA all pieces of the linearly approximated contribution functions must be separately tuned. This means, that the training parameter collection must have input values in every $[a_{i,k}, a_{i,k+1}]$ interval. SNN is a special case of FLA where only one interval is used on each input as mentioned. Thus, in this case it is enough if the training parameter collection has values for every inputs.

Consequently FLA needs documents that are richer in the sense of diversity of the frequency of considered words, however, it results in better estimation. In the next section an example will be shown when not all pieces of the contribution functions are trained.

6. Experiments and results

As it was shown SNN is simplier compared to the standard neural network approach, hence, the learning time was much less. Further, SNN requires 400-500 epochs training to achieve a sufficient estimation, instead of 1000 epochs as in [2]. The same experiments have been conducted using SNN, FLA and the standard approach. The figures show the result calculated by the three techniques respectively. The horizontal axis means the set of considered words, while the vertical axis contains the frequency-keyword measure. To see the difference between the results, the points on the diagrams are connected by lines. Points connected by thin lines show the real frequency-keyword measure from whole documents. Points connected by bold lines show the estimated measures. The results for a certain document by the standard and SNN are shown in Figures 3 and 4 and by FLA in fig. 5.

It can be seen in these figures that the estimations by the new methods are much improved compared to the results by the standard network. FLA results in the best estimation.

As it was mentioned, FLA would result in a much improved estimation increasing the number of antecedent sets, if all pieces of the contribution functions (Fig. 1) were trained. We tried to find intervals that were absolutely not reached by any training parameters and to test the estimation we used a new document where most of the frequency measure values were in these intervals. The results can be seen in figure 5. We added some extra training parameters to tune the untrained pieces of the functions as well. The result can be seen in figure 6.

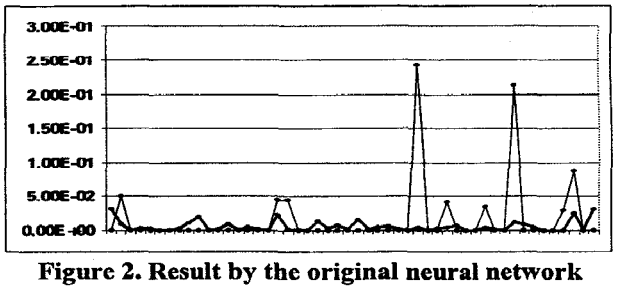

Figure 2. Result by the original neural network

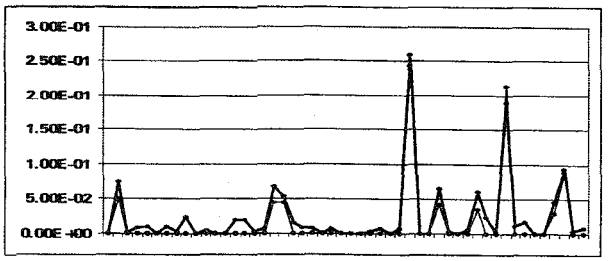

Figure 3. Result by the proposed neural network

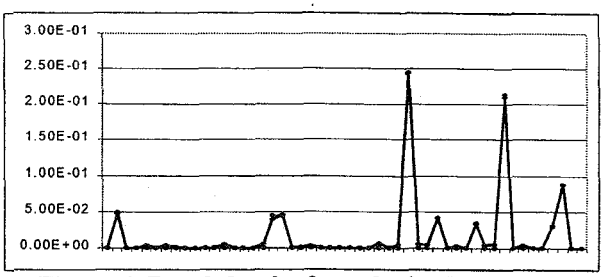

Figure 4. Result by the fuzzy logic algorithm

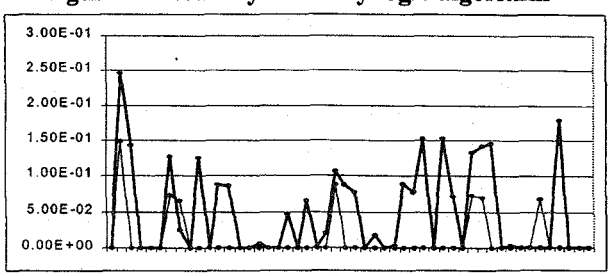

Figure 5.

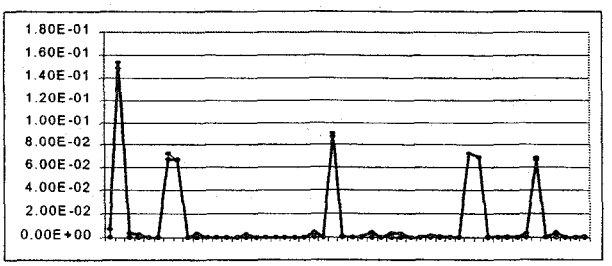

Figure 6.

7. Conclusion

In this paper a FLA and its special case SNN algorithms were introduced. The more fuzzy terms on each universe is used in FLA, the more improved estimation is obtained, however, the more special training parameters and calculation time are required. Using a SNN that is the special case of FLA, namely, where two fuzzy terms are used on each input universe, does not requires specially selected training parameters, however its result is not significantly different from FLA (where the number of input terms is larger than two) in the sense of word-frequency estimation. It was shown that SNN algorithm is much simplified and requies considerably less computation effort than the standard neural network implementations, however, the result is significantly improved.

References

[1] Gedeon, T.D. and Ngu, A.H.H. "Index generation is better than Extarction", *Proceedings NOLTA'93 Int. Conf. Non-Linear Theory and Applications,* pp. 771-777, Hawaii 1993

[2] Bustos, R.A., Gedeon, T.D., "Learning Synonyms and related Concepts in Document Collections" in Alspector, J., Goodman, R. And Brown, T.X. Applications of Neural Networks to Telecommunications 2, Lawrence Erlbaum, 1995, pp. 202-209

[3] Kóczy, L.T., Gedeon, T.D., "Information retrieval by fuzzy relations and hierarchical co-occurence" IETR 97/3 Dept. Imformation Engineering School of Comp. Science and Ing. Univ. of NSW. TR 97/3 Sydney

[4] M.Mizumoto, "Fuzzy controls by Product-sum-gravity method" *Advancement of Fuzzy Theory and Systems in China and Japan,* Eds. Liu and Mizumoto, International Academic Publishers, c1.1-c1.4, 1990.

[5] Gedeon, T.D. and Mital, V., "Information Retrieval using a Neural Network Integrated with hypertext", *Proceedings Int. Conf. Neural networks,* pp 1819-1824, Singapore, 1991.

[6] Rose D. and Belew R., "A Connectionist and symbolic Hybrid for Improving Research", *Int. J. Man Machine Stuies* v. 35, 1991

[7] Blair D.C. *Language and Representation in Information Retrieval*, Amsterdam, Elsevier, 1990.

[8] Blair, DC & Marron, ME "An Evaluation of Retrieval Effectiveness for a Full-text Document Retrieval System," *CACM*, vol. 28, no. 3, pp. 289-299, 1985.

[9] Paice, CD "Constructing Literature Abstracts by Computer: Techniques and Prospects," *Info. Proc. and Management*, vol. 26, no. 1, pp. 171-186, 1990.

[10] Fischer G and Stevens C, "Information access in complex, poorly structured information spaces", *CHI'91 Conference Proceedings*, 1991.

[11]Foltz, PW and Dumais, ST, "Personalized information delivery: an analysis of information filtering methods", *CACM*, vol. 35, no. 12, pp.51-60, 1992.

[12]Goldberg, D, Nichols, D, Oki, BM and Terry, D, "Using collaborative filtering to weave an information tapestry", *CACM*, vol. 35, no. 12, pp.61-70, 1992.

[13] Brookes, C, *grapeVINE: Concepts and Applications*, Office Express Pty. Ltd., 1991.

[14] Salton, G. (1971) *The SMART Retrieval System - Experiment in Automatic Document Processing*, Englewood Cliffs, Prentice-Hall.

[15] Führ, N. & Pfeifer, U. "Combining Model-Oriented and Description-Oriented Approaches for Probabilistic Indexing," *14th International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pp. 46-56, 1991.

[16] Turtle, H, 1995 "Text Retrieval in the Legal World," *Artificial Intelligence and the Law*, vol. 3, pp. 97-142.

[17] Salton, G, 1989 *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison Wesley.