Journal of Computing and Information Technology - CIT 17, 2009, 3, 285–293
doi:10.2498/cit.1001284

Applying the Generalized Dombi Operator Family to the Speech Recognition Task

Gábor Gosztolya[1], József Dombi[2] and András Kocsor[3]

[1]Research Group on Artificial Intelligence of the Hungarian Academy of Sciences
[2]Department of Informatics, University of Szeged
[3]Research Group on Applied Intelligence Non-Profit Company

In the automatic speech recognition (ASR) problem, the task of constructing one word- or sentence-level probability from the available phoneme-level probabilities is a very important one. Here we try to improve the performance of ASR systems by applying operators taken from fuzzy logic which have the sort of properties this problem requires. In this paper we do this by using the Generalized Dombi Operator, which, by its two adjustable parameters and incorporating other well-known fuzzy operators, seems quite suitable. To properly adjust these parameters, we used the public optimization package called Snobfit. The results show that our approach is surprisingly successful: the overall error rate was significantly reduced.

Keywords: speech recognition, triangular norms, Dombi operator family, probability calculation

1. Introduction

In speech recognition the basic problem is to assign the most probable phoneme sequence (i.e. a word or word sequence) to a given speech sample, which also can be viewed as a decision problem. This process can be divided into several parts. First, we extract various features from the input in the signal processing phase. Next, probabilities are assigned to several small chunks which correspond to different phonemes, usually by applying some statistical machine learning method. Then we consider the possible phoneme-sequences and the bounds between them, and calculate one hypothesis-level score based on the small, individual probabilities. After, we search for the most probable hypothesis of all, by applying some search method.

In this paper we shall focus on the third part, i.e. word-level probability aggregation. For this task we will apply the operators called triangular norms taken from the field of fuzzy logic. In the past we carried out some experiments [8, 15, 9] where various well-known and widely-used norms were employed at different levels of the probability calculation tasks. In this study we will describe the results of experiments with the Generalized Dombi Operator [3], which includes several norms from our past experiments as special cases. This time we seek to find the best combination of its parameter values using an optimization package called Snobfit [11].

The structure of this paper is as follows. First we define the speech recognition problem, including its various approaches which frequently appear in the literature. Second, we define some basic terms of fuzzy logic, and identify some subtasks of the speech recognition problem where fuzzy functions can and will be applied. Third, we present the Generalized Dombi Operator. Lastly, we describe the test environment and the test process, then analyse and discuss the test results.

2. The Speech Recognition Problem

In speech recognition problems we have a speech signal represented by a series of observations

$A = a_1 a_2 \dots a_t$, and a set of possible phoneme sequences (words or word sequences, referred to simply as words from now on) that will be denoted by W. Our task is to find the word which fits this speech signal in an optimal way, i.e. a word $\hat{w} \in W$ such as

$$\hat{w} = \arg\max_{w \in W} P(w|A). \quad (1)$$

The *discriminative* approach of speech recognition makes use of Eq. (1). We, however, first apply Bayes' theorem, where we have

$$\hat{w} = \arg\max_{w \in W} \frac{P(A|w) \cdot P(w)}{P(A)}. \quad (2)$$

Further, noting the fact that $P(A)$ is the same for all $w \in W$, we have that

$$\hat{w} = \arg\max_{w \in W} P(A|w)P(w). \quad (3)$$

Throughout this paper we will follow the generative approach using Eq. (3) [12]. We should also remark here that $P(w)$ is usually supplied by some *language model*, which is not the main focus of this paper, hence we will just assume that it is given, and concentrate on $P(A|w)$.

Now we would like to somehow decompose the probabilities represented in Eq. (3) into smaller probability factors that can be handled more easily. To do this, first let the word w be the phoneme sequence $o_1 \dots o_n$, where o_j is the jth phoneme of this word w. Next we will assign segments of the speech signal to each phoneme in order. For this reason let $A_1, \dots, A_n$ be these non-overlapping segments of the original observation series $A = a_1 \dots a_t$, where $A_j = a_{t_{j-1}} \dots a_{t_j}$, $j \in \{1, \dots, n\}$. An A_j segment can also be defined by its start and end times and can be denoted by $[t_{j-1}, t_j]$; also, the set of these time indices can be represented as a vector $S = [t_0, t_1, \dots, t_{n-1}, t_n]$ with a length of $n+1$ ($1 = t_0 < \dots < t_n = t$), which will be referred to as a segmentation.

Now we will make the conventional assumption that the phonemes in a word are independent. By doing this, the probability $P(A|w)$ can then be obtained from $P(A_1|o_1), \dots, P(A_n|o_n)$ in some way (which usually means multiplication). The $P(A_j|o_j)$ (or $P([t_{j-1}, t_j]|o_j)$) values in effect measure how well the A_j segment models the o_j phoneme. There are actually several ways these probability values can be calculated.

2.1. Operators in Probability Calculation

This general way of calculating the probability of a given word with a given segmentation can, of course, be made more specific, but to do this we will need some more theory. Here g_1 will denote the method which assigns a probability for a phoneme on a particular segment (i.e. $P(A_j|o_j)$). It may seem surprising at first, but in some cases (as in our current experiments) it actually does involve an operator. Now let g_2 be the operator which is used when we want to calculate the word-level probability from the given phoneme-level probability scores; i.e.

$$P(A|w) = g_2(P(A_1|o_1), \dots, P(A_n|o_n)). \quad (4)$$

In theory, we do not place any restrictions on this g_2 (except, of course, the trivial ones that it should take any number of arguments, and its result should lie between 0 and 1), but we would like to emphasize here that its default value is the multiplication operator; thus, by default, $P(A|w)$ is calculated as the product of the $P(A_i|o_i)$ values.

The right choice of g_1 and g_2 is vital for a good speech recognition system. This is because we will choose the word (i.e. phoneme-sequence) and segmentation (i.e. time indices representing the bounds between the phonemes) pair which has the highest probability on the given speech signal; so, by varying these operators, the most probable word could also change. Therefore the accuracy (the ratio of correct words over the total number of words) of the system can be seriously affected. Naturally, we are interested in finding the optimal pair of operators.

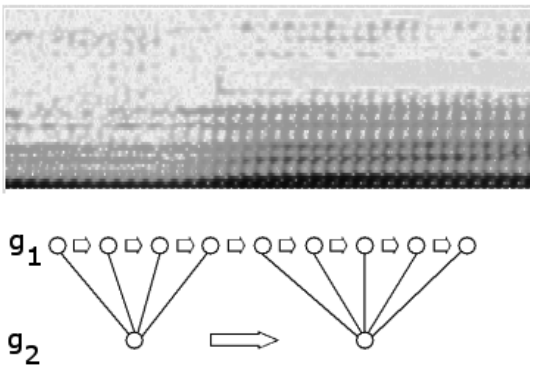

Figure 1. Operators g_1 and g_2 in the frame-based model.

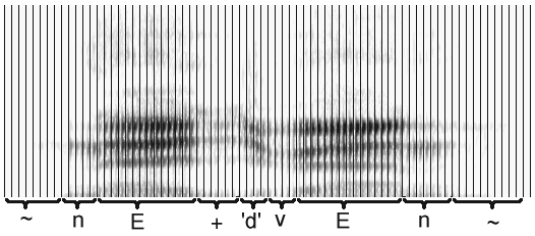

Figure 2. The scheme of frame-based speech recognition. The phonemes of the Hungarian word "negyven" (meaning forty) are assigned to the spectral representation of an utterance, frame-by-frame.

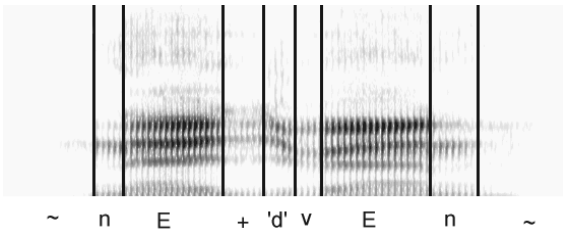

Figure 3. The scheme of segment-based speech recognition, with the same utterance and phonemes as in Figure 2. Each phoneme is assigned to a segment of the spectral representation.

Next we shall discuss the two main approaches and their default g_1 and g_2 operator choices. Note that they will be applied on a given speech sample (A), phoneme sequence $(o_1 o_2 \dots o_j)$ and segmentation $(S = [t_0, t_1, \dots, t_n])$, thus these now will be regarded as fixed values.

2.2. Frame-based Speech Recognition

The well-known *Hidden Markov Model* (*HMM*) [19] is essentially a frame-based method, which means that it handles the speech signal on a frame by frame basis. Hence first we compute each small frame of the speech signal to see how well it represents the corresponding phoneme (i.e. the $P(a_l|o_j)$ values). This is usually done by applying a *Gaussian Mixture Model* (*GMM*) [5], which models the distribution of frames using a set of Gaussian curves. Then these frame-level values of one A_j segment (and thus one o_j phoneme) are aggregated to one probability by the g_1 operator; i.e.

$$g_1(A_j|o_j) = g_1([t_{j-1}, t_j]|o_j) = \prod_{l=t_{j-1}}^{t_j} c_{o_j} \cdot P(a_l|o_j), \tag{5}$$

where the c_{o_j} value is a *state transition probability* $(0 \leq c_{o_j} \leq 1)$. The scheme of the frame-based model is shown in Figure 2. We should mention here that instead of GMMs, Artificial Neural Networks (*ANN*) [1] or any other machine learning algorithm that can be used for density estimation is also viable. This provides a way of creating model hybrids.

As for the g_2 operator, it is simply defined by

$$P(A|w) = P(A_n|o_n) \prod_{j=1}^{n-1} (1 - c_{o_j}) P(A_j|o_j). \tag{6}$$

Since several authors reported that the state transition values have practically no influence on performance, they can be omitted – which is just what we will do in the following [2, 21], which leads to the simpler formula

$$P(A_j|o_j) = \prod_{l=t_{j-1}}^{t_j} P(a_l|o_j) \tag{7}$$

and

$$P(A|w) = \prod_{j=1}^{n} P(A_j|o_j). \tag{8}$$

The way the g_1 and g_2 operators work in the frame-based model is shown in Figure 1.

Our own framework (which was used for testing) can behave both in a segment-based way and in a frame-based manner, but the experiments described in this paper were done in a frame-based setting. We used ANN, but practically any machine learning could have been applied. The choice of method should have no influence on the outcome of the experiments performed.

2.3. Segment-based Speech Recognition

In the segment-based speech recognition approach – as in the SUMMIT system of MIT [7] and in our OASIS Speech Laboratory [17] operating in a segment-based configuration – g_1 will usually be the direct output of some machine learning algorithm like GMMs or ANN. To be able to do this, we will need features which describe the whole $[t_{j-1}, t_j]$ segment; these are typically averages of basic features for the whole segment, or on some specific part of it, like

the beginning, middle or the end. Sometimes a length normalization is also used on the output of the machine learning method, which means raising it to the $(t_j - t_{j-1})th$ power. Figure 3 illustrates this approach.

As for g_2, among the many possibilities, the conventional choice is simply to multiply the probabilities, but using other operators could be beneficial to the overall performance [8].

3. Triangular Norms

As we have seen, we could change the operators applied both in g_1 and in g_2, but it is not clear which operator could work well in these cases. To narrow the range of possible operators, we used two criteria. One was that the behaviour of the operators should be easily modifiable, which is best done by operators with (one or more) parameters. The other was that since the default operator applied was the multiplication operator, we sought to use operators that were "multiplication-like". This latter criterion is indeed not a well-defined one, but its meaning is intuitively clear: the operators we apply should behave in a similar way to the multiplication operator. The *triangular norms* are standard operators of fuzzy logic [4, 14], and they fulfil both requirements, thus we chose to work with them in our study.

Now, following the work of Fodor [6], we will define some basic terms.

Definition. A function $T : [0,1]^2 \to [0,1]$ is a *triangular norm* (*t-norm*) if and only if it satisfies the following conditions:

$$\begin{array}{ll} \text{(T1)} & T(1,x) = x \qquad \text{for all } x \in [0,1]. \\ \text{(T2)} & T(x,y) = T(y,x) \quad \text{for all } x,y \in [0,1]. \\ \text{(T3)} & T(x,y) \le T(u,v) \\ & \qquad \text{for any } 0 \le x \le u \le 1, \\ & \qquad 0 \le y \le v \le 1. \\ \text{(T4)} & T(x,T(y,z)) = T(T(x,y),z) \\ & \qquad \text{for all } x,y,z \in [0,1]. \end{array}$$

(T2) means that a triangular norm must be commutative, (T3) states that T is nondecreasing in both arguments, while (T4) means that T is associative. Note that (T1) and (T3) together imply that $T(0,x) = 0$ for all $x \in [0,1]$. It is easy to see that such a t-norm is also the product operator. Now we need to make some more definitions.

Definition. A t-norm T is said to be

(a) *continuous* if T as a function is continuous on the unit interval (i.e. on $[0,1]$);

(b) *Archimedean* if $T(x,x) < x$ for all $x \in (0,1)$.

These two properties allow us to represent such triangular norms in a more general way:

Theorem. A t-norm T is continuous and Archimedean if and only if there exists a strictly decreasing and continuous function $f : [0,1] \to [0,\infty]$ with $f(1) = 0$ such that

$$T(x,y) = f^{(-1)}(f(x) + f(y)),$$

where $f^{(-1)}$ is the pseudo-inverse of f defined by

$$f^{(-1)}(x) = \begin{cases} f^{-1}(x) & \text{if } x \le f(0) \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, this representation is unique up to a positive multiplicative constant [6]. If the t-norm is strictly monotonously increasing, then

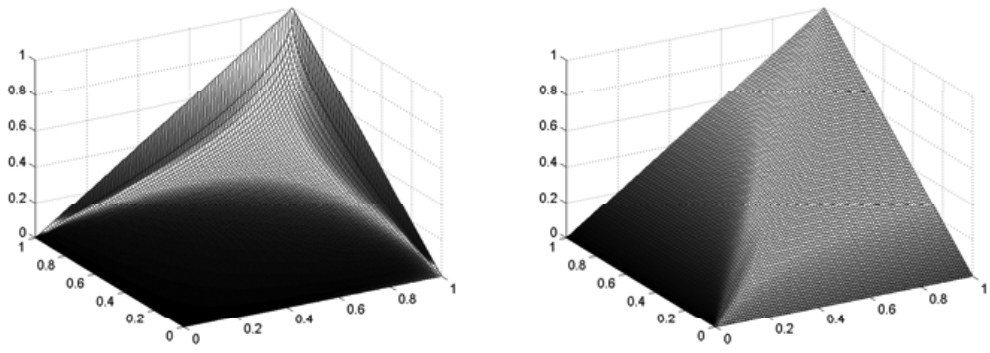

Figure 4. The Dombi triangular norm with $\lambda = 0.3$ and $\lambda = 3$.

the pseudo-inverse function $f^{(-1)}$ is the normal inverse. In our study we will deal with this case as in practical applications these kinds of operators are typically used. We say that a continuous Archimedean t-norm T is *generated by* f if T has such a representation f. In this case f is said to be an *additive generator* of T.

Note that, due to its basic properties, a triangular norm can easily be extended into an n-ary operator as

$$T(x_1,\ldots,x_n) = T(\ldots T(T(x_1,x_2),x_3),\ldots,x_n). \tag{9}$$

Now we will turn to the application of such operators in the speech recognition task.

3.1. Application of Triangular Norms in Speech Recognition

When applying different triangular norms in the hypothesis probability calculation subtask, we have two straightforward possibilities. First, we can use them to create phoneme-level scores from the frame-level probabilities, i.e. as g_1. Obviously, this is only possible in a frame-based model. Second, we can also apply them when aggregating the phoneme-level probabilities to word-, sentence- or hypothesis-level values, i.e. as g_2. In our study we chose the first option, but our previous results [16] imply that usually there is no need to alter g_2 after optimizing g_1.

The next problem which arises concerns the properties we expect from a function applied either as g_1 or as g_2. First, we should expect a slightly different phoneme probability to affect the hypothesis probability by a correspondingly small amount. In other words, there should be no sudden jumps between these kinds of hypotheses. This suggests that this operator should be continuous. On the other hand, if it satisfies the Archimedean property, then the longer a sentence is, the closer the result (and hence the probability of the word sequence pronounced) is to zero, which is also desirable. Another reason for anticipating these properties is that our default operator (the product operator) is also both continuous and Archimedean, thus an operator with these properties fulfils the requirement of being "multiplication-like".

In the past we experimented with various operators in the speech recognition problem. First we tested the root-power mean operator to aggregate the costs of phonemes on word-level values [8]. Then we applied different families of triangular norms for the same task [9]. After, we used uninorms in a segment-based framework [15]. This time, however, we would like to try out some more general operator rather than just experiment with a few triangular norm families. This is because the latter choice has the drawback that perhaps our experiments do not employ the best operator for this task, so it is desirable that the operator being tested should include as many operator types as possible. This led us to choose the Generalized Dombi Operator and use it for g_1. Thus, we will employ the formula

$$P(A_j|o_j) = T(P(a_{t_{j-1}}|o_j),\ldots,P(a_{t_j}|o_j)), \tag{10}$$

where T is an n-ary extension of a triangular norm, and we will look for a suitable T for this task. On the other hand, g_2 remained its default value (multiplication), hence we will apply

$$P(A|w) = \prod_{i=1}^{n} P(A_j|o_j). \tag{11}$$

4. The Generalized Dombi Operator Family

Dombi recently introduced a generalized family of triangular norms and their pairs, the triangular conorms [3]. Now we are interested in the triangular norm case, which means that the operators can be represented in the following way:

$$T_{GD,\gamma}^{(\alpha)} = \frac{1}{1+D_\gamma(x)} \qquad \alpha > 0 \tag{12}$$

where

$$D_\gamma(x) = \left(\frac{1}{\gamma}\left(\prod_{i=1}^{n}\left(1+\gamma\left(\frac{1-x_i}{x_i}\right)^{\alpha}\right)-1\right)\right)^{1/\alpha} \tag{13}$$

and $\gamma > 0$. The corresponding *triangular conorm*, which is the fuzzy generalization of the addition operator, can be also derived from Eq. (12) with $\alpha < 0$.

The generator function of the Generalized Dombi triangular norm is

$$f_c(x) = \ln\left(1+\gamma\left(\frac{1-x}{x}\right)^{\alpha}\right), \tag{14}$$

with the same α and γ values as before, i.e. $\alpha > 0$, $\gamma > 0$. The additive generator function for the corresponding triangular conorm is the same, but $\alpha < 0$.

What makes this operator rather attractive and potentially useful is that it includes many well-known and widely-used triangular norm families as special cases, and among them there are some (like the Dombi and the Hamacher operator families) which proved useful in our earlier tests [8]. Table 1 lists the operator type and the corresponding γ and α values.

Type of Operator	Value of γ	Value of α
Dombi	$\gamma = 0$	$\alpha > 0$
Hamacher	$0 < \gamma < \infty$	$\alpha = 1$
Einstein	$\gamma = 2$	$\alpha = 1$
Product	$\gamma = 1$	$\alpha = 1$
Drastic	$\gamma = \infty$	$\alpha > 0$

Table 1. Some special cases of the Generalized Dombi Operator.

5. Experiments

At this point, having defined the general problem and the operator used, we turn to the testing part. As the technical details could be of importance, we will now elaborate on them.

5.1. The Test Database

To prepare the test environment, two steps have to be performed. First, in each case, we have to train a speaker-independent phoneme classifier to supply the values for $P(a_l|o_j)$ as we will follow the frame-based approach. (In a segment-based context, of course, the same would be true for the $P(A_j|o_j)$ values.) Then, if we want to carry out tests on a sentence recognition task – as we do now –, we should also somehow assign probability values to the words. The module which generates these $P(w)$ probabilities is called the *language model*. However, if we were to perform isolated word recognition where only one word is identified at a time, we could, of course, omit this step.

The Artificial Neural Networks phoneme recognition module was trained on a large, general database to guarantee speaker-independency. 332 people of various ages spoke 12 sentences and 12 words each, which were recorded with different microphones on different computers and sound cards [23]. The features were the standard 13 MFCC values along with their derivatives and the derivatives of the derivatives (MFCC+Δ + $\Delta\Delta$) [10]. This way the phoneme classifier was not only speaker-independent, but it could also be used in any context.

In the next step we combined this phoneme classification method with a simple language model. Here the sentences spoken were restricted to those of medical reports. The language model was a simple word 2-gram; i.e. the probability of the next word depended only of the last word spoken, and it is calculated via a statistical analysis of texts in a similar field. We carried out this investigation on all the available reports (nearly 9,000), which contained 2,500 different words in 95,000 sentences.

The combination of these two methods were then tested on 150 randomly selected sentences, which were of course taken from the same type of texts, namely medical reports. (Otherwise, the language model, which models sentences belonging to this domain, would not have been of much use.) These sentences were tested one after the other, which was done in our OASIS speech recognition framework [17]. This system was originally designed to perform segment-based speech recognition, but due to its flexibility and module-based nature, frame-based recognition is also possible.

5.2. Measurement of Performance

The performance of a speech recognition system can be easily measured on word recognition tasks: we only have to compute the ratio of the correctly recognized and the tested words. However, we cannot use this method on sentence recognition, because just one badly identified word would ruin the whole sentence. We cannot compare the two sentences word for word either, because one incorrectly inserted or omitted word would also corrupt the calculated performance ratio. For this reason, usually the edit distance of the two sentences (the original and the resultant) is calculated; that is, we construct the resulting sentence from the original by using the following operations: inserting and deleting words, and replacing one word

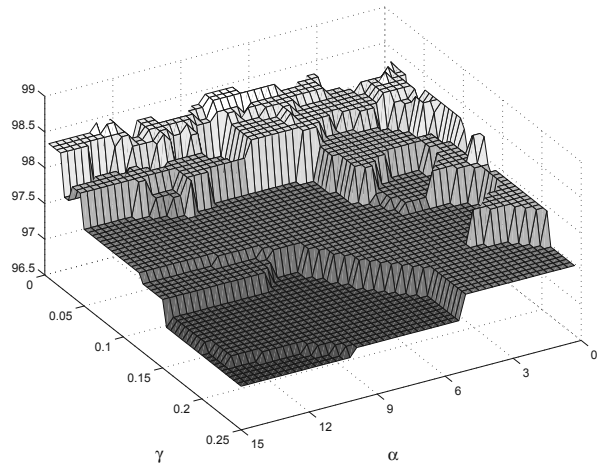

Figure 5. The accuracy values for the α and γ parameters.

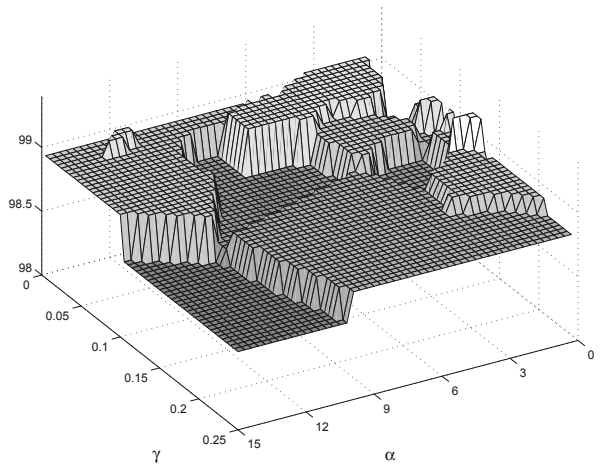

Figure 6. The correctness values for the α and γ parameters.

with another one. These operations have some cost (in our case the common values of 3, 3 and 4, respectively), and then we pick an operation set with the lowest cost. Now we can calculate the following measures:

$$\text{Correctness} = \frac{N-S-D}{N} \tag{15}$$

and

$$\text{Accuracy} = \frac{N-S-D-I}{N}, \tag{16}$$

where N is the total number of words in all the original sentences, S is the number of substitutions, D is the number of deletions and I is the number of insertions. Under these circumstances, the baseline values were 96.76% and 98.38% (accuracy and correctness, respectively), which was probably due to the large number of words and the simple nature of the language model.

5.3. Utilizing Triangular Norms

As we follow the frame-based approach this time, first the phoneme-frame pair probability estimates (the $P(a_l|o_j)$ values) are determined by applying some machine-learning method, for which we utilized ANNs. The g_1 operator, which aggregates phoneme-level scores from these frame-level values, was changed to the n-ary extension of the Generalized Dombi Operator, hence

$$P(A_j|o_j) = T^{(\alpha)}_{GD,\gamma}(P(a_{t_{j-1}}|o_j),\ldots,P(a_{t_j}|o_j)), \tag{17}$$

while g_2 remained the default product operator. But in order to achieve satisfactory results, we have to set α and γ.

5.4. Optimizing the Parameters

The optimization process is generally straightforward if only one parameter needs to be set. This was the case in our previous studies, where operators with only one parameter were tested. Every triangular norm family had a certain range for a given task where it works satisfactorily; all we needed to do was first determine this interval with preliminary tests (i.e. by trying out various parameter values manually) and then explore it with a sufficiently low step size. For triangular norm families with several parameters, however, this approach is not a practical one as it is unlikely that we can find the optimum by varying the parameters one by one. In this case we will need a global optimization method to find a global optimum, and since we utilize the Generalized Dombi Operator family this time with its two parameters, this is what we shall do in the following.

5.5. The SNOBFIT Package

For optimization we chose the SNOBFIT (Stable Noisy Optimization by Branch and FIT) [11] package. It is available as a Matlab [18] package, and it is an optimization system designed for noisy functions which have parameters that

vary between fixed bounds. The ranges of the γ and α parameters were fixed by preliminary tests by trying out a number of value pairs manually and rejecting the badly-performing regions. The function value was calculated from the accuracy value; since SNOBFIT seeks to minimize this function value, we calculated the reciprocal rate of accuracy. Another reason for choosing SNOBFIT is that the calculation of this function involves the execution of another application (i.e. our OASIS speech recognition system), which can also be done within it.

6. Results

In Figure 5 and Figure 6 the test results, i.e. the recognition scores (accuracy and correctness, respectively) can be seen for the different α and γ pairs. For the sake of clarity we show the resultant accuracy values on a three dimensional surface. Since the SNOBFIT algorithm performed tests on discrete $\alpha - \gamma$ pairs, and tested only promising values, there could be areas where no test was made at all. For these points we used the accuracy value of the closest test case.

The first and very interesting observation is that very few configurations led to a decrease in the accuracy score. The reason for this might simply be due to computer arithmetic (underflowing). The second observation is that the accuracy value improved quite significantly: from 96.76% to 98.49% when $\gamma = 0.1$ and $\alpha = 0.7$ and in the neighbourhood of this point. It actually corresponds to a relative error reduction (RER) of 53.4% (i.e. the error rate was reduced from 3.24% to 1.51%, thus by 53.4%), which is a surprisingly good result. The correctness value also increased from 98.38% to 98.95%, which is a slightly lower, but nevertheless impressive relative error reduction of 35.19%. This difference is most likely due to the fact that we optimized the accuracy value. (But since it is the more important of the two, it is best done this way). The percentage ratio of correct sentences also rose from 92.66% to 94.66%, meaning a 27.25% reduction in the sentence-level error value.

We would like to emphasize that it is not the actual optimal α and γ values that are important, nor the exact accuracy and correctness relative error reduction values. These most likely vary from one speech recognition system to another, perhaps even from task to task. What we find most interesting is that by applying the Generalized Dombi Operator in g_1 and g_2, and selecting the proper parameter values for the actual task, the performance of the speech recognition system can be significantly improved by this amount (although slightly different error reduction scores could arise under different conditions). Moreover, it is an improvement which requires practically no additional running time since in speech recognition the signal processing, phoneme-identifying and searching tasks are so CPU demanding, that this makes the calculation of a few fuzzy operators practically negligible.

7. Conclusions

The application of different fuzzy operators in various decision tasks has a long history [13, 22, 20]. In this study we applied them in speech recognition, where the way we construct a word- or sentence-level score based on the probability of smaller regions being phonemes is important. We used the Generalized Dombi Operator because of its flexibility and its generality (i.e. incorporating several widely-used fuzzy operators). The results justified our approach: with the right parameter settings our system was able to increase its word-level accuracy from 96.76% to 98.49% on a sentence-recognition task of medical reports, which meant a relative error reduction of 53.4%.

8. Acknowledgment

A. Kocsor was supported by the János Bolyai fellowship of the Hungarian Academy of Sciences. This research was partially supported by the TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency.

References

[1] C. BISHOP, *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[2] H. BOURLAND, H. HERMANSKY AND N. MORGAN, Towards increasing speech recognition error rates. *Speech Communication*, 18 (1996), pp. 205–231.

[3] J. DOMBI, Towards a general class of operators for fuzzy systems. *IEEE Transaction on Fuzzy Systems*, 16(2) (2008), pp. 477–484.

[4] D. DUBOIS AND H. PRADE, *Fundamentals of Fuzzy Sets*. Kluwer Academic Publisher, 2000.

[5] R. DUDA AND P. HART, *Pattern Classification and Scene Analysis*. Wiley & Sons, New York, 1973.

[6] J. FODOR AND M. ROUBENS, *Fuzzy Preference Modelling and Multicriteria Decision Support*. Kluwer Academic Publisher, 1994.

[7] J. GLASS, J. CHANG AND M. MCCANDLESS, A probabilistic framework for features-based speech recognition. In *Proceedings of the 1996 International Conference on Spoken Language Processing*, pages 2277–2280, Philadelphia, PA, 1996.

[8] G. GOSZTOLYA AND A. KOCSOR, Aggregation operators and hypothesis space reductions in speech recognition. In *Proceedings of the 2004 Conference on Text, Speech and Dialogue*, pages 315–322, Brno, Czech Republic, 2004.

[9] G. GOSZTOLYA AND A. KOCSOR, Using triangular norms in a segment-based automatic speech recognition system. *International Journal of Information Technology and Intelligent Computing*, 1(3), (2006).

[10] X. HUANG, A. ACERO AND H.-W. HON, *Spoken Language Processing*. Prentice Hall, 2001.

[11] W. HUYER AND A. NEUMAIER, Snobfit – stable noisy optimization by branch and fit. *ACM Transactions on Mathematical Software*, 35(2), (200), pp. 1–25.

[12] F. JELINEK, *Statistical Methods for Speech Recognition*. The MIT Press, 1997.

[13] N. KASABOV, R. KOZMA, R. KILGOUR, M. LAWS, M. J. WATTS, A. R. GRAY, AND J. G. TAYLOR, *Speech Data Analysis and Recognition Using Fuzzy Neural Networks and Self-Organised Maps*, pages 241–263. Physica Verlag, 1999.

[14] E. KLEMENT, R. MESIAR, AND E. PAP, *Triangular Norms*. Kluwer Academic Publisher, 2000.

[15] A. KOCSOR AND G. GOSZTOLYA, Application of full reinforcement aggregation operators in speech recognition. In *Proceedings of the 2006 Conference of Recent Advances in Soft Computing (RASC)*, Canterbury, UK, 2006.

[16] A. KOCSOR AND G. GOSZTOLYA, The use of speed-up techniques for a speech recognizer system. *The International Journal of Speech Technology*, 9(3-4), (2006), pp. 95–107.

[17] A. KOCSOR, L. TÓTH AND J. A. KUBA, An overview of the OASIS speech recognition project. In *Proceedings of the 1999 International Conference on Applied Informatics*, Eger-Noszvaj, Hungary, 1999.

[18] MATHWORKS, Matlab, 1984-2008. `http://www.mathworks.com`.

[19] L. RABINER AND B.-H. JUANG, *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[20] C. SCHRUMPF, M. LARSON AND S. EICKELER, Syllable-based language models in speech recognition for English spoken document retrieval. In *Proceedings of AVIVDiLib*, pages 196–205, Cortona, Italy, 2005.

[21] G. TÓTH AND A. KOCSOR, Explicit duration modelling in HMM/ANN hybrids. In *Proceedings of the 2005 Conference on Text, Speech and Dialogue*, pages 310–317, Karlovy Vary, Czech Republic, 2005.

[22] D. TRAN AND M. WAGNER, Fuzzy Expectation-Maximisation algorithm for speech and speaker recognition. In *Proceedings of NAFIPS*, pages 421–425, 1999.

[23] K. VICSI, A. KOCSOR, C. TELEKI AND L. TÓTH, Beszédadatbázis irodai számítógép-felhasználói környezetben (in Hungarian). In *Proceedings of MSZNY 2004*, pages 315–318, Szeged, Hungary, 2004.

Received: May, 2008
Revised: July, 2008
Accepted: February, 2009

Contact address:
Gábor Gosztolya
Research Group on Artificial Intelligence
Hungarian Academy of Sciences and
University of Szeged
Aradi vértanúk tere 1., 6720 Szeged, Hungary
e-mail: `ggabor@inf.u-szeged.hu`

GÁBOR GOSZTOLYA received his MSc degree in Computer Science in 2001 from the University of Szeged. Currently he is a Ph.D. student at the Research Group on Artificial Intelligence, Hungarian Academy of Sciences and University of Szeged. His current research interests include various aspects of speech recognition and fuzzy logic applications.

JÓZSEF DOMBI received his Ph.D. degree in Quantum Chemistry in 1977 from the University of Szeged, Hungary. He has been researching fuzzy set theory and fuzzy systems since 1978, and he received the Candidate of Mathematical Sciences degree from the Hungarian Scientific Academy in 1994. Currently he is an associate professor at the Department of Informatics of the University of Szeged. He is a member of the International Fuzzy Systems Association, founder and former head of Cygron Ltd., the developer of DataScope which won the Software of the Year award in 1999 at COMDEX, Las Vegas. His research interests are primarily the area of fundamentals of fuzzy sets, including membership functions, modifiers and operators, and also the area of multicriteria decision making.

ANDRÁS KOCSOR is a senior researcher at the Research Group on Applied Intelligence Non-profit Company in Szeged, Hungary. His current research interests include kernel-based machine learning, similarity measures, speech recognition, speech synthesis, inequalities, and mathematics techniques applied in artificial intelligence.