On best fit T-norms in speech recognition

Gábor Gosztolya
Research Group on Artificial Intelligence
of the Hungarian Academy of Sciences
and the University of Szeged
Szeged, Hungary
Email: ggabor@inf.u-szeged.hu

László L. Stachó
Bolyai Institute
University of Szeged
Szeged, Hungary
Email: stacho@math.u-szeged.hu

***Abstract*—We generalize the model of automatic speech recognition (ASR) based on maximization of products of probability likelihoods of speech frame-phoneme correspondences by applying strict t-norms. We formulate it as a minimization problem in terms of the logarithmic generator of strict t-norms and investigate the experimental solutions in cases of piecewise linear logarithmic generators. The performance of the best fit t-norms found in this manner for a database used in earlier papers with classical t-norms is proved to be essentially superior than the results there.**

I. Introduction

Most speech recognition systems rely on assigning a sequence of parameters with values between 0 and 1 to short time speech segments, that show the probability likelihood of the given segment to correspond phonemes. On the other hand, we are given a dictionary of strings of phonemes (words or complete sentences) which should be compared with the matrix consisting of the mentioned probability likelihoods of the speech signal unit in order to determine which sequence of items in our dictionary should be taken as the best guess for the speech signal. The traditional strategies try to identify parts of the speech signal which may correspond to a given dictionary item. Given a sequence of shorter consecutive speech segments which are regarded as guessed performances of phonemes, one can determine fitness parameters for dictionary words consisting of as many phonemes as the number of the given speech segments. (This procedure can be easily extended for words having less phonemes that the number of segments by stretching some phonemes.) This fitness parameter determination mostly done on the basis of the above mentioned speech-phoneme probability likelihood values, namely by simply taking their products.

In this paper we shall be primarily interested in improving the seemingly arbitrary step of replacing the simple products of speech-phoneme probability likelihood values by their T-norms. In several earlier articles [6], [5] we investigated the effect of applying some widespread T-norms in this decision procedure. In particular the parameters of the family of generalized Dombi norms were optimized for this purpose. But though these norms cover several classical ones, they are far from being of a general character and it is natural to ask if there are even better ones among the family of general T-norms. It is well-known that any continuous Archimedean T-norm is of equivalent order topologically either to the product norm or to the Lukasiewicz norm [7]. Here we shall be concerned with finding the best fit T-norm. Taking into account the fact that the classical Lukasiewicz norm produces a very low performance in this context, it can be expected that even its order topological equivalents would be less suitable for this kind of application, thus we will focus on product-equivalent T-norms.

The work was supported by the NKTH grant of Jedlik Ányos R&D Programme 2007 of the Hungarian government (codename TUDORKA7).

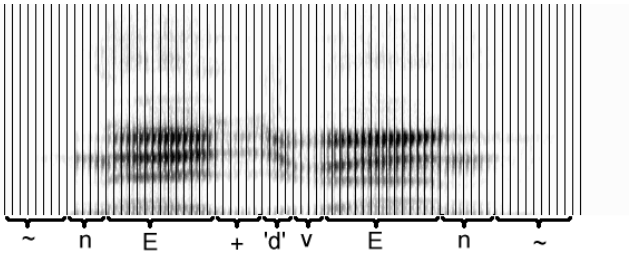

Fig. 1. An example utterance of the Hungarian word ”negyven” (meaning forty) portrayed in its spectral form, and divided to small, equal-sized parts (frames).

II. The Speech Recognition Process

In the speech recognition problem the task is to assign the correct word from a dictionary to a given speech signal. But without a priori knowledge it is not possible to tell for sure whether a given word is the correct one or not, so in practice we aim for the word which is the ”most fitting” one. This should be done strictly without human interaction, but the procedure of course should result in the correct word in as many cases as possible. There is a common way of doing it, which we will describe in the following, and then define a way for improving it (i.e. making it supply the correct words in more cases than it did before).

The speech signal, after some signal processing steps, is defined as a series of equal-sized vectors that describe significant information for t short time, equal-sized speech segments called *frames*. For an example, see Figure 1. (Note that this paper follows a frame-based description of the speech recognition problem. For details, see [**?**].) Now we will consider the set of possible phonemes $o_1, o_2, \ldots, o_N$, and use a

	Frames								
	1	2	3	4	5	6	7	$\cdots$	t
$o_1 =$ a	0.3	0.2	0.2	0.3	0.4	0.2	0.3	$\cdots$	0.1
$o_2 =$ b	0.1	0.3	0.2	0.1	0.2	0.1	0.1	$\cdots$	0.2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$o_{25} =$ n	0.2	0.2	0.1	0.3	0.4	0.3	0.5	$\cdots$	0.1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$o_{48} =$ z	0.1	0.1	0.2	0.1	0.0	0.2	0.1	$\cdots$	0.1
$o_{49} = \sim$	0.6	0.6	0.5	0.6	0.4	0.4	0.2	$\cdots$	0.5

TABLE I
EXAMPLE OF PROBABILITY ESTIMATES FOR PHONEMES.

	Frames								
	1	2	3	4	5	6	7	$\cdots$	t
$o_1 =$ a	0.3	0.2	0.2	0.3	0.4	0.2	0.3	$\cdots$	0.1
$o_2 =$ b	0.1	0.3	0.2	0.1	0.2	0.1	0.1	$\cdots$	0.2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$o_{25} =$ n	0.2	0.2	0.1	0.3	0.4	**0.3**	**0.5**	$\cdots$	0.1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$o_{48} =$ z	0.1	0.1	0.2	0.1	0.0	0.2	0.1	$\cdots$	0.1
$o_{49} = \sim$	**0.6**	**0.6**	**0.5**	**0.6**	**0.4**	0.4	0.2	$\cdots$	**0.5**
$p_{\nu(5,.)}$	**0.6**	**0.6**	**0.5**	**0.6**	**0.4**	**0.3**	**0.5**	$\cdots$	**0.5**

TABLE II
THE $p_{\nu,\tau}$ VALUES FOR A GUESS OF THE WORD "NEGYVEN", WHICH HAS SILENCE AS FRAMES 1 TO 5, THE PHONEME "N" STARTING FROM FRAME 6, AND SILENCE FRAMES AT THE END.

standard procedure to calculate the *frame-phoneme probability matrix*

$$\mathbf{P} = \big[p_{i\tau} : \ 1 \le i \le N, \ 1 \le \tau \le t\big],$$
$$p_{i\tau} = \big[\text{prob. likelihood of phoneme } i \text{ corresponding to frame } \tau\big].$$

This step is usually done by some machine learning method such as the Gaussian Mixture Model (GMM) [4] or Artificial Neural Networks (ANNs) [1]. As an illustration let us take the pronounced word *negyven* (forty in Hungarian) with $t = 100$ and $N = 49$. As a starting step with the GMM procedure we get the matrix $\mathbf{P}$ as in Table I, for instance $p_{1,1} = 0.3, \quad p_{1,2} = 0.2, \ \ldots \ , \ p_{49,100} = 0.5.$

Then we turn to the word set. For this purpose we have a dictionary which contains all the possible words which are to be matched against the speech signal. Each word (or even word sequences) can be treated simply as a sequence of phonemes already transcribed manually or by some algorithm, hence we can treat them as phoneme-sequences. A sample dictionary would be one like this:

$\sim$ a b b a $\sim$	for the word "abba"
$\sim$ n E + 'd' v E n $\sim$	for our pronounced "negyven"
$\vdots$	$\vdots$.

The crucial step of a recognition procedure at this stage is to associate *fitness values* for *word-pronunciation guesses*. These guesses are words from the dictionary, with phonemes stretched so that one phoneme is assigned to each frame. E.g.

$\sim\sim\sim$aaaaaaabbbbbbbbbbbbaaaaaaa$\sim\sim \cdots \sim$
for the word "abba"
$\sim\sim\sim$aaaaabbbbbbaaaaaa$\sim\sim \cdots \sim$
for "abba" pronounced shorter
$\vdots$
$\sim\sim\sim\sim\sim$nnnnEEEEEE++'d"d"d'vvvEEEEnnnn$\sim \cdots \sim$
for our pronounced "negyven"
$\vdots$.

There could of course be many such guesses, but the number of investigated guesses should be drastically reduced using well-known simple heuristical methods not being in the focus of this paper. Next, let

$$\nu(n,\tau) := \big[\text{ the index of the } \tau\text{-th phoneme in guess } n\big].$$

In our example $\nu(1,1) = \nu(1,2) = \nu(1,3) = 49, \nu(1,4) = \nu(1,5) = \cdots = \nu(1,10) = 1, \ \nu(1,11) = \cdots = \nu(1,21) = 2, \nu(1,22) = \cdots = \nu(1,28) = 1, \ \nu(1,29) = \cdots = \nu(1,100) = 49$ because our first guess begins with 3 consecutive silent frames (" $\sim$ " $= o_{49}$) followed by 7 "a" frames ($= o_1$) etc. For a sample such ν values for our example "negyven" see Table II above.

In course of the classical procedure the fitness value F_n for guess n is simply calculated by taking the product of the probability likelihood values of its phonemes as

$$F_n = \prod_{\tau=1}^{t} p_{\nu(n,\tau),\tau} \ , \tag{1}$$

and our final guess should be the one where $\max_n F_n$ is taken. Heuristically, taking the product of probability likelihoods for the fitness value corresponds to assuming high scale independence between the consecutive frames in the speech signal. Though even this standard approach has proved to be quite successful, it is natural to expect that it can be improved further by replacing the product in the formula of F_n with a more general binary operation on the interval $[0,1]$ which is commutative, associative and increasing with unit 1 and sink 0. These operations are exactly the so-called T-norms of fuzzy logics and they are used to calculate the certainties (probability likelihood) of an element belonging to the intersection of two fuzzy sets from the certainties of its belonging to the intersecting sets. Thus we will now give a brief discussion of T-norms.

III. STRICT TRIANGULAR NORMS

A *strict triangular norm* is a binary operation $T : [0,1]^2 \to [0,1]$ such that

(a) $T(x,y) = T(y,x)$, $T(T(x,y),z) = T(x,T(y,z))$,
(b) $T(0,x) = 0$, $T(1,x) = x$,
(c) $T(x_1,y) < T(x_2,y)$ for all $x_1 < x_2, y \neq 0$.

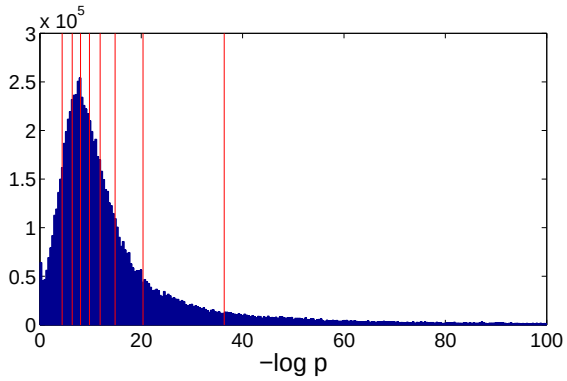

Fig. 2. A histogram of the $-\log p$ values appearing during a standard speech recognition process using multiplication, on the interval $[0, 100]$

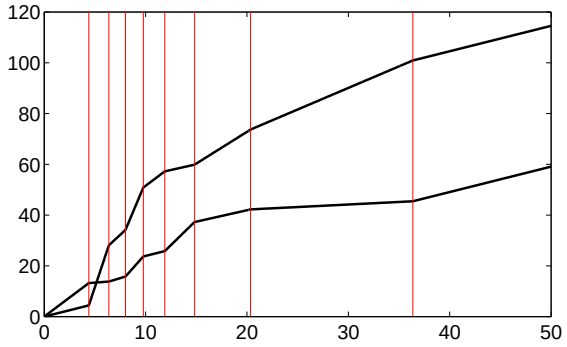

Fig. 3. Two sample logarithmic generator functions with control points corresponding to Figure 2

Recall [3], [10] that all strict continuous t-norms admit the representation

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

with some suitable strictly decreasing continuous function $f : [0,\ 1] \to [0, \infty]$ such that $f(0) = \infty$ and $f(1) = 0$. The function f above is said to be an *additive generator* of T. Any strictly decreasing surjective function $f : [0, 1] \to [0, \infty]$ is the additive generator of some strict t-norm, and two additive generators f_1 and f_2 give rise to the same t-norm if and only if they are positive multiples of each other.

In the classical approach (i.e. Eq. (1)) the fitness values are calculated with a strict t-norm $T(x, y) = xy$ corresponding to the additive generator $f(x) = -\log x$. For numerical reasons we consider the equivalent minimization problem

$$\tilde{F_n} = \sum_{\tau=1}^{t}(-\log p_{\nu(n,\tau),\tau}) \to \text{ MIN in } n$$

instead of $F_n = \prod_{\tau=1}^{t} p_{\nu(n,\tau),\tau} \to$ MAX. in n. Thus, in general, when we replace xy with an arbitrary strict t-norm with generator f, it is also convenient to introduce the *logarithmic generator function* $\phi(x) = f(e^{-x})$ and rewrite the general maximization problem

$$F_n = f^{-1}\Big(\sum_{\tau=1}^{t} f(p_{\nu(n,\tau),\tau})\Big) \to \text{ MAX in } n$$

to the equivalent minimization

$$\tilde{F_n} = \sum_{\tau=1}^{t} \phi(-\log p_{\nu(n,\tau),\tau})) \to \text{ MIN in } n.$$

It is important to notice that the family of all strict t-norms with piecewise linear logarithmic generator $\phi : [0, \infty] \to [0, \infty]$ with finitely many breakpoints and such that $\lim_{x\to\infty} \phi'(x) = 1$ is dense in the family of all strict t-norms with respect to the topology of uniform convergence. The proof is just a standard compactness argument.

IV. Choice Strategy for Logarithmic Generators

Henceforth let $\phi = \phi^{m_1,\ldots,m_n}_{a_1,\ldots,a_n} : [0, \infty] \to [0, \infty]$ be the piecewise linear, strictly increasing function with break points $0 = a_0 < a_1 < a_2, \ldots, a_n < a_{n+1} = \infty$ and with steepness values $m_1, m_2, \ldots, m_n > 0$ respectively $m_{n+1} = \lim_{x\to\infty} \phi'(x) = 1$. That is,

$$\phi(x) = (x - a_j)m_{j+1} + \sum_{i=1}^{j}(a_j - a_{j-1})m_j,\ a_j \le x < a_{j+1}.$$

This representation has several advantages. If the control points are fixed, a function ϕ can be described by the vector of the n steepness values, making it easy to optimize. On the other hand, the function ϕ is unique up to a positive multiplicative constant; now, by setting m_{n+1} to 1, we fix exactly one of these equivalent representations. Furthermore, we have the possibility of placing these control point a_j-s to values where they represent the problem we are currently modelling, as accurately as possible.

This way, by fixing all the a_j values, and every other possible settings of the speech recognition environment, this problem can be simplified to that of a maximization one in an n-dimensional space. That is, given a vector $\overline{m} = (m_1, m_2, \ldots, m_n)$, we seek to maximize the accuracy of the speech recognition system as a function of this $\overline{m}$ vector (i.e. $Acc(\overline{m})$).

A. The Choice of Control Points

The only task left now is to accurately place the control points. For this we suggest a simple test: let us perform an ordinary speech recognition process with the default operator, i.e. with $T(x, y) = xy$, $f(x) = -\log x$. During this test let us note which x and y values are passed to the operator (and thus to the generator function f). Owing to the commutativity property we do not need to distinguish between the two arguments, i.e. x and y. Next, calculate a histogram of the $-\log$ of recorded values, i.e. for each value note how many times it has appeared. Finally, to actually assign the n control points, divide this histogram into $n + 1$ equal-sized parts; the control points will be the borders between these regions.

This way, during a typical usage, roughly the same number of evaluations will fall into each part of the function ϕ between two adjacent control points, so that each steepness value will have about as importance as the others.

In our case it means a speech recognition test using multiplication, i.e. f is $-\log x$. The resulting histogram of appearing $-\log x$ values and a sample list of control points can be seen in Figure 2, while some possible logarithmic generator ϕ functions are shown on Figure 3.

Finally, the actual function f (and thus, the triangular norm T) can be easily calculated from ϕ. Of course it will not be piecewise linear, but piecewise an exponential function with a negative index. It will be continuous, but not smooth, i.e. its derivative will be a discontinuous function (except, of course, the case where every steepness value is 1, which is exactly the product case).

V. Experiments and Results

Having our problem and solution defined, we now turn to testing. We will describe the speech recognition environment, the actual definition of the function to be optimized, and the software package we used for the optimization process. Finally we will present our results and draw our conclusions.

A. The Speech Recognition Environment

First let us describe the environment this method of t-norm modelling was tested in. All testing was done in our OASIS speech recognition framework, which, due to its module-based structure and script-based execution, was quite suitable for this kind of experimental testing [11].

The probability estimates for a frame being a particular phoneme were supplied by an Artificial Neural Networks (ANNs) method [1] with a classic structure of one hidden layer. The feature vectors (the a_i values) were also ones commonly used for speech recognition: the 13 Mel-frequency Cepstral Coefficients (or MFCC) were calculated, along with their derivatives, and the derivatives of the derivatives (MFCC + Δ + $\Delta\Delta$ for short), making 39 features in total [8].

The ANNs were trained on a large, general database. 332 people of various ages spoke 12 sentences and 12 words each, which were recorded with different microphones on different computers and sound cards [13]. This way a speaker-independent classifier was created, which can be used in practically any situation.

The tests were done not on simple words, but on whole sentences taken from the field of medical reports. In similar cases it is common to have some sort of language model; in our case a simple word 2-gram was used, i.e. the likeliness of a word was only decided by considering it and the previous word (based on a statistical investigation of similar texts). The tests were finally run on 150 randomly selected sentences, one after the other.

B. Measurements 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 words over 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 on words; that is, we construct the resulting sentence from the original by using the following operations: inserting and deleting words, and replacing one word 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:

$$Correctness = \frac{N - S - D}{N} \tag{2}$$

and

$$Accuracy = \frac{N - S - D - I}{N}, \tag{3}$$

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 is probably due to the large number of words and the simple nature of the language model. Besides the word-level correctness and accuracy scores, we calculated the number of correctly recognized whole sentences, which appeared to be 92.66% (i.e. 139 correct sentences out of a total of 150).

C. Setting the Logarithmic Generator Function

As mentioned earlier, we set the control points of the logarithmic generator function by running a standard speech recognition test, and then calculated the histogram of the values appearing there. The points were then placed to the values between $n + 1$ equal-sized regions. We carried out experiments with $n = 8$ and $n = 16$. For the steepness values, as ϕ is a strictly monotonously increasing function, we can say that $m_j > 0$. On the other hand, there is no sure upper bound; but since $m_{n+1} = 1$, we thought that $m_j \leq 10$ would be sufficient. Thus we looked for a point in an n-dimensional hypercube, namely $(0, 10]^n$, which results in a maximal function value.

D. The Snobfit Package

Since we are modelling the generator function as a multi-parameter function, we definitely need a global optimization method. We chose the Snobfit (Stable Noisy Optimization by Branch and FIT) [9] package for this task. It is available as a Matlab 6 [12] package, and it is an optimization system designed for noisy functions which have parameters that vary between fixed bounds. The ranges of the steepness parameters

were fixed between 0 and 10, and 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), and this operation is also supported.

E. Results

Table III shows the best performances of each method used. Beside the baseline values of the product operator and the results of our new modelling approach, the performance of two other t-norms are shown for reference: that of the Dombi triangular norm family, and that of the generalized Dombi operator family [2]. The parameter value of the former one was determined via a simple sequence of tests, where the latter one, having two parameters, had to be also optimized with Snobfit [6], [5].

Method	Accuracy	Correctness	Sentences
Product (baseline)	96.76%	98.38%	92.66%
Dombi t-norm	97.57%	98.84%	93.33%
Generalized Dombi operator	98.49%	98.95%	94.66%
Modelled t-norm, $n = 8$	98.27%	98.84%	94.00%
Modelled t-norm, $n = 16$	98.84%	99.19%	96.00%

TABLE III
THE BEST ACCURACY AND CORRECTNESS VALUES FOR THE DIFFERENT METHODS TESTED, AND THE RATIO OF CORRECT SENTENCES.

It can be seen that beyond the baseline values (which mean the usage of the product operator), significant improvements can be attained. Even by using the somewhat simple Dombi operator can reduce the error rates significantly. Using the generalized Dombi operator leads to even better results, but its application is more complicated because it has two parameters instead of only one. However, with the logarithmic generator function no further difficulties arise, and it could lead to a better performance (i.e. higher accuracy and correctness values). In addition, the ratio of correct sentences can be increased. To obtain this result, however, there should be a sufficient number of control points in order to provide the t-norm with enough freedom to fit the problem it is applied to. This could be the reason why our proposed method with $n = 8$ could not attain the performance of the generalized Dombi operator. (It still performed much better than the product, however, especially regarding the accuracy score, which it was optimized for.)

But with $n = 16$, it was able to outperform all the methods tested in all measurements. Quite clearly, when we have enough freedom to optimize the generator function (and thus, the behaviour of the triangular norm it generates), it can fit to the particular problem even better than the well-performing classical t-norm families we tested. The usage of too many control points, however, may lead to a case where the search space is so high dimensional that finding an optimum can be hard or almost impossible. Fortunately with $n = 16$ this was not the case, so this choice of n seems to be a good compromise between easy optimization and robustness in our case.

Lastly, we would like to stress that the usage of the logarithmic generator function to model t-norms is not restricted to the field of speech recognition. Although our tests were limited to this field, we see no reason why this idea should not work in any field that makes use of triangular norms. Its application may require, of course, some small modification to find a good value of n.

VI. CONCLUSION

One possible application domain for triangular norms is that of calculating word probability estimations from probability estimations of phonemes for small speech segments. By finding an appropriate operator for this problem we can significantly improve the performance of the speech recognition system. Many triangular norm families have one or more parameters to fine-tune them in this task, but the question which naturally arises is that are they flexible enough to adequately fit to the given problem? To answer this we introduced the *logarithmic generator function*, and by optimizing it for piecewise linear terms with 16 break points on a sample of 150 sentences with 865 words, we could indeed significantly increase the precision of this speech recognition procedure over the results achieved with classical and Dombi t-norms. This positive outcome of our experiments raises the interest for testing out best fit t-norm with a much lager database. It is also worth to remark that our optimization method is not limited to the setting of speech recognition, actually applications in improving some fuzzy control algorithms can also be expected.

July 14, 2008

REFERENCES

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

[2] J. Dombi. Towards a universal fuzzy concept: General operators. *Accepted for IEEE Transaction on Fuzzy Systems*, 2007.

[3] D. Dubois and H. Prade. *Fundamentals of Fuzzy Sets*. Kluwer Academic Publisher, 2000.

[4] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley & Sons, New York, 1973.

[5] G. Gosztolya, J. Dombi, and A. Kocsor. Applying the Generalized Dombi Operator family to the speech recognition task. *Submitted for CIT*, 2008.

[6] 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.

[7] P. Hájek. *Metamathematics of Fuzzy Logic*. Kluwer Academic Publishers, 1998.

[8] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing*. Prentice Hall, 2001.

[9] W. Huyer and A. Neumaier. Snobfit - stable noisy optimization by branch and fit. citeseer.ist.psu.edu/681619.html.

[10] E. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer Academic Publisher, 2000.

[11] 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.

[12] Mathworks. Matlab, 1984-2008. http://www.mathworks.com.

[13] 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.