# Using the Logarithmic Generator Function in the Spoken Term Detection Task*

Gábor Gosztolya

MTA-SZTE Research Group on Artificial Intelligence
of the Hungarian Academy of Sciences and University of Szeged
H-6720 Szeged, Tisza Lajos krt. 103., Hungary
ggabor@inf.u-szeged.hu

**Abstract.** Spoken term detection is a task in artificial intelligence where user-entered keywords are to be looked for in a huge audio database. In one common approach the recordings are first converted into phoneme-sequences, and the actual search is performed in this space. During search, instead of performing the default multiplication of basic phoneme operation probabilities, applying a triangular norm can significantly improve system accuracy. We used an application-oriented method for triangular norm representation and tuning, namely the logarithmic generator function. In practice this proved to be quite successful and led to a relative error reduction score of 16%.

**Keywords:** triangular norms, additive generator function, artificial intelligence, speech processing, spoken term detection, keyword spotting.

## 1 Introduction

Among the range of fuzzy functions, *triangular norms* (or *t-norms*) [7] have a significant number of successful applications in the literature, especially in artificial intelligence (AI) problems such as image enhancement [4], image blending [13], classifier combination [3], speech recognition [9], and multimodal biometrics [14]. What is common among these AI tasks, and what makes them a good area for applying t-norms, is that they usually rely on aggregating lower-level probabilities (outputs of single classifiers, phoneme probabilities of short excerpts of speech, confidence scores of different biometrical identifier systems etc.). The standard approach for this aggregation is to simply calculate the product of these individual probability values (*naive Bayes* approach), relying on the assumption that these components are independent. While this assumption leads to a nice and elegant mathematical formulation and also behaves well in practice, in most cases it is clearly false, which calls for the use of other operators. However, these

* This publication is supported by the European Union and co-funded by the European Social Fund. Project title: Telemedicine-focused research activities in the fields of mathematics, informatics and medical sciences. Project number: TÁMOP-4.2.2.A-11/1/KONV-2012-0073.

operators still have to express an AND-like relation of the arguments. Triangular norms are just the conjunctive operators of fuzzy logics, and they are an ideal choice for such tasks.

In this paper we will focus on an AI task, that of spoken term detection (STD, sometimes also referred to as *keyword spotting* or KWS), which is a quite recent topic within speech technology. It seeks to provide a way to search for user-entered keywords in a huge archive of audio recordings. Recent approaches [27,28] view this task in a dictionary-independent way, where search is performed only by relying on the acoustic model and using only general language information (e.g. probability values of consecutive phoneme pairs or triplets). This rules out the approach of simply performing automatic speech recognition (ASR [25]) on the recordings, storing the resulting *word sequence*, and performing a text search in this textual representation, since this approach prevent users from finding words (usually proper nouns) which were not present in the dictionary used in the speech recognition step.

One common approach in STD is to represent the recordings as mere phoneme-sequences, to which the phonemes of the search term are matched one by one. The overall probability of such a phoneme-sequence pairing is usually computed as the product of the individual probability values. In this paper we experimented with triangular norms when performing this aggregation; among the wide range of possible t-norms we chose an application-oriented representation.

In this paper we will describe the STD problem, focusing on the approach using phoneme-sequences. Then we will describe the t-norm representation chosen (the logarithmic generator function), present the test results, and finally analyze them.

## 2 The Spoken Term Detection Task

In the spoken term detection task we seek to find the user-entered natural language expressions (*terms* or *keywords*) in an audio database (the set of *recordings*). An STD method returns a list of *hits*, each of which contains the point of occurrence, the term found, and a probability value that can be used to rank the hits. In contrast to other information retrieval tasks, in STD the order of the hits does not matter; the probability value of the returned hits is only used to filter the hit list further by using a decision threshold, keeping just the more probable elements.

In STD, a user expects a quick response for his input, thus we have to scan hours of recordings in a few seconds (or less). To achieve this, the task is usually separated into two distinct parts. In the first one, steps requiring intensive computation are performed without knowing the actual search term, resulting in some intermediate representation. Then, when the user enters the keyword(s), a (quick) search is performed in this representation. We will focus on the approach where the intermediate representation is the most probable phoneme sequence, since it permits a very quick search while still retaining good accuracy [20].

The most probable phoneme sequence for each recording is usually generated by some standard speech recognition technique. Then, for a term $w$ with the phonemes $w_1, w_2, \ldots, w_n$, we look for all the non-overlapping phoneme sequences ($L$) for which

$$P(w|L) \geq P_{\min}, \tag{1}$$

where $P_{\min}$ is a threshold set previously. Making the standard assumption that the successive phonemes are independent, we get

$$P(w|L) = \prod_{i=1}^{n} P(w_i|l_i) \geq P_{\min}, \tag{2}$$

where $l_1, \ldots, l_n$ are the phonemes of the phoneme sequence $L$. To compensate for errors in the phoneme sequence representations, phoneme insertions, deletions and substitutions are allowed. This means that $w_i$ or $l_i$ can be empty ($\lambda$), so

$$P(w|L) = \prod_{i=1}^{m} P(w_i|l_i) \geq P_{\min}, \tag{3}$$

where by omitting the $w_i = \lambda$ values from the sequence $w_1, \ldots, w_m$ we get the term $w$, and without the $l_i = \lambda$ values $l_1, \ldots, l_m$ forms $L$. $P(w_i|\lambda)$ represents the probability of deleting phoneme $w_i$ (if $w_i \neq \lambda$), $P(\lambda|l_i)$ means the probability of inserting phoneme $l_i$ (if $l_i \neq \lambda$), while $P(w_i|l_i)$ is the probability of substituting $w_i$ for $l_i$ in the case where neither $w_i$ nor $l_i$ is $\lambda$ (but it may be that $w_i = l_i$). The optimal pairs can be found by calculating the edit (or Levenshtein) distance [22]. The probability values of the phoneme operations can be computed from the errors of the phoneme recognizer: after performing phoneme classification on recordings with known real phonetic transcriptions, the probability values of phoneme insertions, deletions and substitutions can be readily calculated by comparing the resulting phoneme sequences to ground truth ones (i.e. from the confusion matrix [21,10]).

Note that in equations (2) and (3) we made the assumption that the consecutive phonemes are independent, which allowed us to decompose $P(w|L)$ into a product of lower-level probability values. This assumption is clearly false owing to the continuous motion of the vocal chords, the tongue and the mouth [31], so we can replace product with other operators as long as they behave well in practice. As triangular norms also represent AND-like relations of values in the range $[0, 1]$, which is just what we need for combining probability values of phoneme operations (insertions, deletions and substitutions), we may expect them to work well in this task. Furthermore, several norms (e.g. [5,26,1,6]) have one or more parameters, allowing us to fine-tune them to the actual problem. For these reasons, we will apply t-norms in the STD task.

## 3 The Logarithmic Generator Function

One advantage of using triangular norms is their tunability: they can be adapted to the requirements of the given problem. With respect to this, however, there could be great differences among various t-norm families depending on how the range of triangular norms they contain matches the ideal performance needed for our actual application [11]. On the basis of our earlier findings [9,12] it is usually better to concentrate on the additive generator function $f$ [26,17], since we have plenty of room to adjust it to suit our actual needs. This can be viewed as triangular norm construction [18,8], with respect to the criterion that the applied triangular norm representation must be easy to handle.

Recall that a strict, continuous and Archimedean triangular norm $T$ can be written in the form

$$T(x,y) = f^{-1}\big(f(x) + f(y)\big), \tag{4}$$

where $f$ is the *additive generator* of $T$, and it is a continuous, strictly decreasing function on the interval $[0,1]$; $f(0) = \infty$ and $f(1) = 0$. Moreover, for a given $T$, $f$ is unique up to a scalar factor, so the triangular norm applied can also be represented by its generator function. If we could find a suitable way to model this function $f$, we could fine-tune its behaviour to suit our needs. To achieve an optimal performance we have to find a flexible yet simple representation, preferably one which is application-oriented.

The additive generator is widely examined in the literature (e.g. [19,23,18]). However, for an actual application we need an application-oriented approach instead of a theory-oriented solution, as we have to pay attention also to computer arithmetics (like the ability of avoiding underflowing, being able to easily handle values in a different order of magnitude, etc.). Due to these reasons we chose the logarithmic generator function for triangular norm representation, which we will describe next.

### 3.1 The Logarithmic Generator Function

To understand the logic of the logarithmic generator function [12], we should first consider its application context. In a typical case we have a number of probability estimates as input $(p_1, p_2, \ldots, p_k)$, and a t-norm $T$; and we need to calculate

$$T^k(p_1, p_2, \ldots, p_k) = T(\ldots T(T(p_1, p_2), p_3), \ldots, p_k). \tag{5}$$

Now using the transcript $T(x,y) = f^{-1}(f(x) + f(y))$ we have that

$$T^k(p_1, p_2, \ldots, p_k) = f^{-1}\Big(\sum_{i=1}^{k} f(p_i)\Big). \tag{6}$$

In our environment, and in most artificial intelligence tasks, to avoid numerical underflowing, instead of a probability value $p$ we use the cost value $c = -\log p$. This step also implies that we use cost addition instead of probability multiplication, and perform (aggregated) cost minimization instead of probability maximization. The triangular norms, however, work only on probability values. To overcome this difficulty, first we incorporate this conversion into Eq. (6), i.e. we will use

$$-\log\Big(f^{-1}\Big(\sum_{i=1}^{k} f(e^{-c_i})\Big)\Big). \tag{7}$$

It is straightforward to include the calculation of the negative exponential into $f$; hence, the logarithmic generator function is defined as

$$\phi(x) = f(e^{-x}). \tag{8}$$

Now we can write

$$\phi^{-1}\Big(\sum_{i=1}^{k}\phi(c_i)\Big) = -\log\Big(f^{-1}\Big(\sum_{i=1}^{k} f(e^{-c_i})\Big)\Big) \tag{9}$$

$$= -\log T(e^{-c_1}, e^{-c_2}, \ldots, e^{-c_k}), \tag{10}$$

so using the logarithmic generator function $\phi(x)$ in exactly the same way as we used the additive generator function $f(x)$ will lead to a calculation of the same triangular norm $T$, only with the corresponding cost values instead of the probabilities both as arguments and as the result. As $f(x) : [0, 1] \to [0, \infty)$ was a strictly decreasing function with $f(1) = 0$, the logarithmic generator function $\phi(x) : [0, \infty] \to [0, \infty]$ is strictly increasing, and $\phi(0) = 0$. The additive generator function is unique up to a multiplicative constant for any given $T$ t-norm, so the same is true for the logarithmic generator function.

## 3.2 Representing the Logarithmic Generator Function

Now we will turn to modeling this logarithmic generator function. Almost any representation could be used for this task; we chose to model it with a piecewise linear one for two basic reasons. First, it is quite simple to handle: both $\phi$ and $\phi^{-1}$ can be implemented very easily. Second, it is a very flexible representation: the family of all strict t-norms with a piecewise linear logarithmic generator $\phi : [0, \infty] \to [0, \infty]$ with finitely many breakpoints, 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. A proof of this just involves a standard compactness argument.

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 on the domain as $0 = a_0 < a_1 < \ldots < a_N < a_{N+1} = \infty$ and with positive steepness values $m_1 < \ldots < m_N$, respectively, and $m_{N+1} = \lim_{x\to\infty} \phi'(x) = 1$. That is,

$$\phi(x) = (x - a_j)m_{j+1} + \sum_{i=1}^{j}(a_j - a_{j-1})m_j, \qquad a_j \le x < a_{j+1}. \tag{11}$$

If the $a_i$ control points are fixed, $\phi$ can be described by a vector of $N$ steepness values, making it easy to optimize. Furthermore, the function $\phi$ is unique up to a positive multiplicative constant; by setting $m_{N+1}$ to 1, we fix exactly one of these equivalent representations. The actual function $f$ (and hence, the triangular norm $T$) can be easily calculated from $\phi$, being a piecewise exponential function with $N + 1$ negative exponents. It will be continuous, but not smooth (except when $m_1 = \ldots = m_N = 1$, which is just the product case).

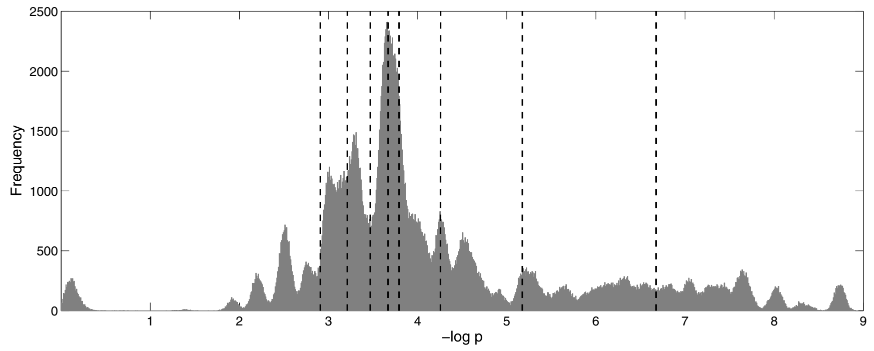

**Fig. 1.** A (smoothed) histogram of the $-\log p$ values encountered during performing STD, and the suggested positioning of the $N = 8$ control points

This way, by keeping all the $a_j$ values fixed, this problem can be simplified to that of a maximization task in an $N$-dimensional space: we seek to maximize the accuracy of the spoken term detection system as a function of $\mathbf{m}$. As for the choice of the control points, we have the possibility to set them at values where they represent our problem as accurately as possible. Since it is also nontrivial, next we will present a method for control point assignment.

## 3.3 The Choice of Control Points

Optimizing the logarithmic generator function means performing a search in an $N$-dimensional vector space. To aid this search process we should avoid the presence of irrelevant or redundant dimensions, so we should try to give each one the same importance. The main idea behind the general method introduced for this purpose in [12] is to create statistics of the values occurring during use, i.e. note which $x$ and $y$ values are passed to the $T(x, y)$ operator (and thus to the generator function $f$). Owing to the commutative property we do not need to distinguish between the two arguments $x$ and $y$. Next, we calculate a histogram of the $-\log$ of recorded values: for each value we note how many times it appears. Afterwards, we divide this histogram into $N + 1$ parts with equal-sized areas: the control points will be the borders between these regions (see Fig. 1). This way about the same number of evaluations will fall into each region between two adjacent control points, making each steepness value (roughly) equally important. An advantage of this method is that it is quite general regarding the actual task, since it requires only a statistic of appearing cost values, and it also has only parameter ($N$).

Now we have presented the logarithmic generator function, which allows us to represent and fine-tune a triangular norm in an application-oriented manner. We have also described a general methodology to fit it into a given problem by positioning the $a_i$ control points. Next, we will focus on the actual application.

## 4 Experiments and Results

Having defined the problem and the logarithmic generator function, we turn to the testing part: we introduce the evaluation methodology, the testing environment and the way of testing, then present and analyze the test results.

### 4.1 The Evaluation Metrics

A Spoken Term Detection system returns a list of hits for a query. Given the correct list of hits, we should rate the performance of the system to compare different configurations. In STD, instead of standard information retrieval metrics such as precision (the ratio of correct hits found to the hits returned) and recall (the ratio of correct hits found to all the correct hits), usually some other, albeit similar measures are used. Here, we will mainly use the Actual Term-Weighted Value (ATWV) [24], which is defined as

$$ATWV = 1 - \frac{1}{T}\sum_{t=1}^{T} \big(P_{Miss}(t) + \beta P_{FA}(t)\big), \tag{12}$$

where $T$ is the number of terms, $P_{Miss}(t)$ is the probability of missing the term $t$ (in fact, the opposite of recall for the term $t$) and $P_{FA}(t)$ is the probability of getting a false alarm. These values are defined as

$$P_{Miss}(t) = 1 - \frac{N_C(t)}{N_T(t)} \qquad \text{and} \qquad P_{FA}(t) = 1 - \frac{N_{FA}(t)}{T_{speech} - N_T(t)}, \tag{13}$$

where $N_C(t)$ is the number of correct hits returned, $N_{FA}(t)$ is the number of false alarms, $N_T(t)$ is the total number of real occurrences of term $t$, and $T_{speech}$ is the duration of recordings in seconds. Usually the penalty factor for false alarms ($\beta$) is set to 1000. A system achieving perfect detection (having precision and recall scores of 100%) has an ATWV score of 100%; a system returning no hits has a score of 0%; while a system which finds all occurrences, but produces 3.6 false alarms for each term and speech hour also has a score of 0% [24]. An older and more permissive metric is the Figure-of-Merit (FOM), which is the mean of recall scores when we allow only $1, 2, \ldots 10$ false alarms per hour per keyword.

Note that although ATWV uses all the hits returned, a threshold value was still used, namely $P_{\min}$ from Eq. (3). A carelessly chosen threshold constant leads to a worse ATWV score than optimal; due to this, usually it is worth calculating *max-ATWV* (or MTWV), which is a (theoretical) upper bound of ATWV, where we take the maximal ATWV score of all $N$-best lists of the hit list returned. It summarizes the performance of the system if the probability threshold $P_{\min}$ has been optimally chosen. Fortunately, the metric FOM does not rely on this threshold value.

**Table 1.** The accuracy values obtained when using different kinds of triangular norms

| | | Development Set | | Test Set | | |
|---|---|---|---|---|---|---|
| T-norm used | N | MTWV | FOM | ATWV | MTWV | FOM |
| Log. gen., optimized for MTWV | 8 | 71.32% | 89.81% | 65.38% | 67.43% | 86.67% |
| | 16 | 68.44% | 88.15% | 64.21% | 67.43% | 87.31% |
| Log. gen., opt. MTWV + FOM | 8 | 70.71% | 91.13% | 69.22% | 69.75% | 88.55% |
| | 16 | 65.25% | 92.79% | 65.86% | 66.03% | 90.11% |
| Product (baseline) | | 57.31% | 91.13% | 63.29% | 63.78% | 89.96% |

## 4.2 The Testing Environment

We used audio recordings of Hungarian news broadcasts taken from 8 different TV channels for testing. The 70 broadcasts were divided into three groups: the first, largest one (about 5 hours long) was used for training purposes. The second part (about 1 hour long) was the *development set*: these recordings were used to fine-tune the t-norm and get the corresponding threshold. The third part was the *test set* (about 2 hours long), used for the final evaluation of system performance. We chose 25 words and expressions as search terms, coming up in the news recordings quite frequently; they varied between 6-16 phonemes (2-6 syllables) in length. The phoneme sequence intermediate representations were produced by Artificial Neural Networks [2] used in two consecutive steps [29], applying the standard MFCC $+\Delta + \Delta\Delta$ feature set [15] with phoneme bigrams as a dictionary-independent language model, using the HTK tool [30].

## 4.3 The Testing Process

To set the control point $a_i$ values, we used the histogram-based method described in Section 3.3. It requires a statistic of the actual probability values, which was obtained in a simple way. Assuming that the distribution of the phonemes of search terms mirror those of the recordings, we calculated the ratio of the occurrence of each phoneme in the training data set. Next, we chose two phonemes according to this distribution, and noted the probability values of deleting the first phoneme, inserting the second one, and replacing the first phoneme by the second one. This process was repeated 100 000 times, some white Gaussian noise was added to the generated values to smooth the resulting discrete values, and we chose the control points based on this histogram.

We performed the optimization of the steepness values by using the freely available Snobfit package [16]. We maximized for just the MTWV metric, and for the MTWV and FOM metric combined, and experimented with $N = 8$ and $N = 16$ control points, which meant a total of 4 tests. We optimized it by performing STD on the development set; the steepness values associated with the optimal score were then evaluated on the test set, using the corresponding threshold value. To ensure stability, we took all the vectors that produced an optimal score on the development set, and calculated their mean for each steepness $m_i$. In the

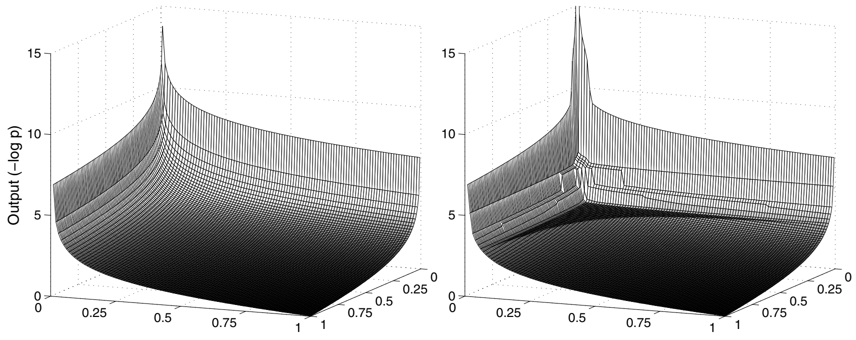

**Fig. 2.** The product t-norm (left) and the optimized t-norm using the logarithmic generator function (right). The x and y axes show the two argument probability values, while the z axis show the resulting cost value (i.e. -log p).

end, we got five scores for each case: MTWV and FOM for the development set, and ATWV (using the threshold value we got on the development set), MTWV (using the optimal threshold for the test set) and FOM for the test set.

## 4.4 Results

Table 1 lists the accuracy scores obtained using the logarithmic generator function. Examining the scores attained on the development set, we can see that all the optimized metric values significantly increased compared to the baseline scores. The settings $N = 16$ produced somewhat worse scores than $N = 8$, which is probably due to the *curse of dimensionality*: the number of tests required increases exponentially with the number of dimensions. Turning to the test set results, we see that in some cases the ATWV score is much lower than MTWV, reflecting threshold instability (i.e. $P_{\min}$ obtained on the development set was not optimal for the test set). In general, error reduction in the test set was not as successful, which could be partly due to *overfitting*: the optimization resulted in a development set-specific t-norm. To avoid this side effect, incorporating other metrics (in our case FOM) into the objective function of optimization seems to be a good idea, as in these cases there were only minor differences in the corresponding MTWV and ATWV scores. This is probably because different evaluation metrics measure the performance of a configuration in a somewhat different manner; in our case ATWV focuses on the top of the hit list, whereas FOM takes less probable hits into account as well. Trying to satisfy both metrics at the same time might result in a more balanced hit list, being better *in general*.

We should also stress that the resulting MTWV and ATWV scores exceeded those of the product norm in every case. Focusing on the case $N = 8$ when we optimized both for MTWV and FOM, we achieved an ATWV score of 69.22%, which, compared to the baseline score of 63.29%, means a relative error reduction score over 16%, this being quite a significant improvement in STD accuracy.

The product t-norm and the best-performing logarithmic generator function can be seen in Figure 2 (where, to emphasize the differences between the two norms, the z axis has a log scale). It can be seen that the two norms are quite different, reflecting the fact that the product operator is suboptimal for this task, and, unlike the logarithmic generator function, it could not be tuned either.

## 5 Conclusions

In a common approach of spoken term detection, user-entered queries are processed by matching their phonemes to the phonemes of recordings one at a time. In this task usually the phoneme operations are assumed to be independent, hence the product of their probabilities is taken; but using a triangular norm instead of multiplication can improve the system accuracy. In this work we applied an application-oriented representation of t-norms, and achieved a significant improvement in system accuracy and resulted in a relative error reduction of 16% this way.

## References

1. Aczél, J., Alsina, C.: Characterizations of some classes of quasilinear functions with applications to triangular norms and to synthesizing judgements. Methods Oper. Res. 48, 3–22 (1984)
2. Bishop, C.: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)
3. Bonissone, P., Goebel, K., Yan, W.: Classifier fusion using triangular norms. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 154–163. Springer, Heidelberg (2004)
4. Deng, G.: A parametric generalized linear system based on the notion of the t-norm. IEEE Transactions on Image Processing 22(7), 2903–2910 (2012)
5. Dombi, J.: A general class of fuzzy operators, the De Morgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. Fuzzy Sets and Systems 8, 149–163 (1982)
6. Dombi, J.: Towards a general class of operators for fuzzy systems. IEEE Transaction on Fuzzy Systems 16(2), 477–484 (2008)
7. Dubois, D., Prade, H.: Fundamentals of Fuzzy Sets. Kluwer (2000)
8. Fodor, J.C.: A remark on constructing t-norms. Fuzzy Sets and Systems 41(2), 195–199 (1991)
9. Gosztolya, G., Dombi, J., Kocsor, A.: Applying the Generalized Dombi Operator family to the speech recognition task. Journal of Computing and Information Technology 17(3), 285–293 (2009)
10. Gosztolya, G., Kocsor, A.: A hierarchical evaluation methodology in speech recognition. Acta Cybernetica 17(2), 213–224 (2005)
11. Gosztolya, G., Kocsor, A.: Using triangular norms in a segment-based automatic speech recognition system. International Journal of Information Technology and Intelligent Computing (IT & IC) (IEEE) 1(3), 487–498 (2006)
12. Gosztolya, G., Stachó, L.L.: Aiming for best fit t-norms in speech recognition. In: Proceedings of SISY (IEEE), Subotica, Serbia, pp. 1–5 (September 2008)

13. Grundland, M., Vohra, R., Williams, G.P., Dodgson, N.A.: Cross dissolve without cross fade: Preserving contrast, color and salience in image compositing. In: Proceedings of Computer Graphics Forum, vol. 25, pp. 577–586 (2006)
14. Hanmandlu, M., Grover, J., Gureja, A., Gupta, H.: Score level fusion of multimodal biometrics using triangular norms. Pattern Recognition Letters 32(14), 1843–1850 (2011)
15. Huang, X., Acero, A., Hon, H.W.: Spoken Language Processing. Prentice Hall (2001)
16. Huyer, W., Neumaier, A.: Snobfit – stable noisy optimization by branch and fit. ACM Transactions on Mathematical Software 35(2), 1–25 (2008)
17. Jenei, S.: On Archimedean triangular norms. Fuzzy Sets and Systems 99(2), 179–186 (1998)
18. Jenei, S.: A general method for constructing left-continuous t-norms. Fuzzy Sets and Systems 136(3), 263–282 (2003)
19. Jenei, S., Pap, E.: Smoothly generated Archimedean approximation of continuous triangular norms. Fuzzy Sets and Systems (Special Issue "Triangular norms") 104, 19–25 (1999)
20. Katsurada, K., Sawada, S., Teshima, S., Iribe, Y., Nitta, T.: Evaluation of fast spoken term detection using a suffix array. In: Proceedings of Interspeech, pp. 909–912 (2011)
21. Kohavi, R., Provost, F.: Glossary of terms. Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process 30(2/3) (February/March 1998)
22. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
23. Ling, C.H.: Representation of associative functions. Publ. Math. Debrecen 12, 189–212 (1965)
24. Pinto, J., Hermansky, H., Szöke, I., Prasanna, S.: Fast approximate spoken term detection from sequence of phonemes. In: Proceedings of SIGIR, Singapore (2008)
25. Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Prentice Hall (1993)
26. Schweizer, B., Sklar, A.: Associative functions and statistical triangle inequalities. Publ. Math. Debrecen 8, 169–186 (1961)
27. Szöke, I., Schwarz, P., Matejka, P., Burget, L., Karafiát, M., Fapso, M., Cernocky, J.: Comparison of keyword spotting approaches for informal continuous speech. In: Proceedings of Interspeech, pp. 633–636 (2005)
28. Tejedor, J., Wang, D., King, S., Frankel, J., Colas, J.: A posterior probability-based system hybridisation and combination for spoken term detection. In: Proceedings of Interspeech, Brighton, UK, pp. 2131–2134 (September 2009)
29. Tóth, L.: A hierarchical, context-dependent Neural Network architecture for improved phone recognition. In: Proceedings of ICASSP, pp. 5040–5043 (2011)
30. Young, S.: The HMM Toolkit (HTK) (software and manual) (1995), `http://htk.eng.cam.ac.uk/`
31. Young, S.: Statistical modelling in continuous speech recognition. In: Proceedings of UAI, Seattle, pp. 562–571 (2001)