University of Szeged
Research Group on Artificial Intelligence

Improving Speed and Accuracy in Automatic Speech Recognition

Summary of the PhD Dissertation

by

Gábor Gosztolya

Advisors:

Prof. Dr. János Csirik
Dr. András Kocsor

Szeged
2010

Introduction

In the problem of speech recognition the task is to assign the correct word or word sequence to a given utterance. Throughout the development of systems and methods for this task there were always two very important factors, regardless of approaches and models applied: accuracy and speed. The accuracy of a speech recognition system is vital as we perform speech recognition to get the correct transcript of the utterance; if it is rarely satisfied, performing speech recognition makes no sense at all. On the other hand, the speed of the speech recognition system is also very important, since we want to get this result within real time; moreover, it is important not to waste resources unnecessarily, thus speed remains an important issue even over and above that of real-time recognition.

The importance of speed and accuracy led myself to make them the central issue of this dissertation. As this work is divided up along these lines, the results will be divided in the same way. In the **first part** we will concentrate on **speeding up** speech recognition, and especially its search subtask (as it is well known that it uses most of the CPU time). In the search subtask a huge number of potential word sequences (and their prefixes, the *hypotheses*) are matched against the utterance. The first general way of speeding up a search is to reduce the number of hypotheses *scanned by the search heuristic*. We focused on improving the multi-stack decoding method [1] by devising a set of smarter limits for the number of hypotheses stored at a given point instead of a simple constant.

Another general way of making a search faster is to reduce the number of *possible* hypotheses by creating new criteria; this way fewer possibilities have to be considered during search. We introduced two solutions for this: the first one was the construction of a *multi-pass search* [27], where several search steps are employed, each being more sophisticated than the previous one, but working only on the resulting hypotheses, to quickly exclude less likely possibilities. Our actual multi-pass algorithm created more compact phoneme groups based on the mistakes of the phoneme classifier. The other solution for making a search faster was to restrict the set of possible positions where phonemes could start or end (called a *segmentation*), for which three novel methods were introduced.

The **second part** of the dissertation considers areas which seek to **improve the performance** of the speech recognition process. The first one is a thorough examination of aggregating probabilities, which is done at two distinct levels: when constructing probabilities of whole segments as phonemes, and when creating probability estimates for a hypothesis from these segment-level probability values. These processes normally use multiplication, derived from the assumption of independence. To find a better-fitting model I applied several similar operators, and I will present a novel method for modelling the fuzzy operators called *triangular norms* from the viewpoint of practical usability. The other problem investigated for improving the accuracy of speech recognition is the so-called *anti-phoneme modelling*. In some approaches of speech recognition we have to distinguish between correct phonemes and any other speech portion like multiple consecutive phonemes or parts of phonemes. It could be done in several ways; apart from the standard ones, I used a general-purpose counter-example generator algorithm for it. To apply it I had to make some improvements to it, but it produced the best results.

Most of the literature treats speech recognition in a frame-based manner, concentrating mainly on Hidden Markov Models. This dissertation, however, contains experiments made both in frame- and segment-based contexts, which are covered by a unified, more general view [29]; it serves as a basis for numerous experiments performed by our team. Further, the experiments were made on a number of databases, that became available during my work; their details are given in the dissertation.

1 Speeding Up the Search Process in Speech Recognition

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. The task is to find the word which fits this speech signal optimally, i.e. a word $\hat{w} \in W$ such as

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

The *discriminative* approach of speech recognition makes use of Eq. (1) [8]. We, however, first apply Bayes' theorem, where we have

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

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

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

We will follow the *generative approach* using Eq. (3) [22]. $P(w)$ is usually supplied by some *language model*, which we be will considered given and we concentrate on $P(A|w)$. This formula can be decomposed into probabilities of smaller units: the word w will be treated as a phoneme sequence $o_1, \dots, o_n$, and A will be divided into non-overlapping segments $A_1, \dots, A_n$, each A_j belonging to phoneme o_j. This way, assuming that the phonemes are independent [8], we can calculate

$$P(A|w) = \prod_{j=1}^{n} P(A_j|o_j). \tag{4}$$

In a *frame-based model* the $P(A_j|o_j)$ values are decomposed further (to the level of frames), whereas in a *segment-based context* they are typically the direct output of the phoneme classifier. In this work we will deal with both frame- and segment-based approaches.

Most search methods applied usually work in a left-to-right manner, placing only one phoneme at a time at the end of the actual prefix of the word. These prefixes are the so-called *hypotheses*, which are object pairs in the form $(o_1 \dots o_j, [A_1, \dots, A_j])$. Concatenating another (allowed) phoneme and a new ending time to a hypothesis is called *extending* it.

The two basic search algorithms we applied were the multi-stack decoding algorithm and the Viterbi beam search method; these can be considered as variants of each other. Both can be implemented by assigning data structures (*stacks*) to every possible place where there could be a bound between phonemes; these stacks (not to be confused with LIFO- or FIFO-style stacks) simply store hypotheses based on their probability. A limited-size stack holds only the most probable ones, and discards the worst ones that exceed its capacity n. The multi-stack decoding algorithm applies limited-sized stacks (n being the only parameter of the method), whereas the Viterbi beam search algorithm keeps just the best hypothesis in the stack, and the ones close to it, which is controlled by the parameter *beam width*. Otherwise, the two search algorithms can be considered as working in the same way.

First the initial hypothesis is put into the stack assigned to the beginning of the speech signal. Then we examine the stacks in increasing time order: for each step we get all the hypotheses from the current stack, extend them in every possible way, and put the generated hypotheses into the stack according to their new ending times. The result of this method is the most probable hypothesis in the last stack having a phoneme sequence belonging to a word from the dictionary [1].

Method combinations	Speed-up ratio on the Numbers Database	Speed-up ratio on the Telephone Database
Viterbi beam search	100.00%	100.00%
multi-stack decoding	68.65%	86.98%
multi-stack + i	241.60%	482.57%
multi-stack + ii	147.77%	358.48%
multi-stack + iii	351.54%	86.98%
multi-stack + iv	166.58%	379.47%
multi-stack + i + ii + iii + iv	756.55%	704.97%

Table 1: Speed-up of recognition of the improvements for the Numbers Database and for the Telephone Database (both isolated word recognition tasks, segment-based cases) while preserving the optimal accuracy. Speed-up is in inverse ratio to actual running speed, compared to the Viterbi beam search.

1.1 Improving the Multi-Stack Decoding Algorithm

The multi-stack decoding search method has a weak point: it uses equal-sized stacks for the entire speech signal, which size is only needed at a few points. (The same is true for the Viterbi beam search with its fixed beam width.) Suggesting techniques to estimate stack sizes more cleverly would reduce the run time of this algorithm without any loss of accuracy. We employed the following improvements:

i) First we combined multi-stack decoding with a Viterbi beam search: we keep only the n best-scoring hypotheses, but also discard those which lie further from the best one than the beam width.

ii) Another idea is based on the observation that at the beginning of the utterance we have very little information to decide which hypothesis should be kept, thus we need bigger stacks than we will need later on. A simple solution for this was applied: the stack size at time t_i will be $s \cdot m^i$, where $0 < m < 1$ and s is the size of the first stack.

iii) Another technique is a small modification of stacks. It is very common to have several hypotheses with matching end times and phoneme sequence (but some earlier phoneme bound is different), when it is enough to keep just the most probable one as further extensions depend only on the ending time and the phoneme sequence, which are the same. The stacks can be easily modified to work this way, but then a time-consuming examination of its content is needed every time a new hypothesis is inserted. So our solution was a simpler one: the stacks can store more of these "same" hypotheses, but when extending a hypothesis from the stack, we first compare its phoneme sequence to the previously examined one. If the phoneme sequences match, this hypothesis is just thrown away.

iv) This idea comes from the observation that we need big stacks only at those segment bounds which correspond to phoneme bounds. So if we could estimate the probability of a given time instance being a bound, we could reduce the size of the scanned hypothesis space. Our solution was to train an ANN for this task on the 13 MFCC Δ features, its output being treated as a probability p. The actual stack size was determined as a function of p: the more complicated method applied an exponential function, while the simpler one was just a step function, used in cases requiring an easy-to-fit function.

These four improvements were tested on two databases, tuned to get as a big speed-up as possible while preserving the original recognition accuracy (see Table 1). It was shown that they can all be combined to give an even greater speed-up. The amount of speed-up achieved reflects the mentioned weak point of the standard search algorithms: their ignorance of the position in the speech signal.

Used passes				d^1 (minimum)		d^2 (average)	
$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{D}_{\min}$	$\mathcal{D}_{\max}$	$\mathcal{D}_{\min}$	$\mathcal{D}_{\max}$
•	○	○	○	100.00%	100.00%	100.00%	100.00%
•	•	○	○	187.81%	188.96%	270.96%	233.32%
•	○	•	○	217.46%	188.23%	–	160.61%
•	○	○	•	–	–	–	**338.70%**
•	•	•	○	162.76%	229.64%	–	172.62%
•	•	○	•	–	–	–	189.22%
•	○	•	•	–	–	–	248.47%
•	•	•	•	–	–	–	194.60%

Table 2: Speed-up results for our multi-pass method for the Children Database (isolated word recognition, segment-based case), compared to multi-stack decoding. "•" means applying the given pass, "○" means omitting it, "–" means that the configuration could not attain the minimal accuracy level.

1.2 Creating a Multi-pass Method

There exist **multi-pass methods** which work in two or more steps: in the first pass the less likely hypotheses are discarded based on some fast-computed condition. Then, in the later passes, just the remaining hypotheses are examined by more complex, reliable evaluations, approximating hypothesis probabilities more precisely [27]. We construct a multi-pass method by combining phonemes into groups: this way several words will be united, which will make the dictionary smaller, and it will result in a faster recognition step. In the later passes only the successful words of an earlier pass will remain, which can be examined more thoroughly.

We clustered the phoneme set by applying a standard clustering algorithm, relying on a distance function d of the original phonemes. At the start every phoneme forms its own cluster; then, at each step, the two closest clusters are united until this minimal distance reaches a limit L. The distance between the two clusters can be either the minimum of the distance between their elements ($\mathcal{D}_{\min}$), or the maximum of them ($\mathcal{D}_{\max}$) [19]. Our method is based on the mistakes of the phoneme classifier which supplies the $P(a_i|o_j)$ (frame-level) or $P(A_j|o_j)$ (segment-level) values: we use its confusion matrix M, where the element $m_{i,j}$ is the number of phonemes belonging to the class ω_j which were recognized as ω_is by the classifier. First it is *normalized* to reflect the ratio of misclassified items; i.e.

$$m'_{i,j} = \frac{m_{i,j}}{\sum_k m_{k,j}}, \quad 1 \le i,j \le K, \tag{5}$$

where K is the number of classes. Then we define the phoneme distance function d based on M', in two variations:

$$d^1_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } m'_{i,j} = m'_{j,i} = 0 \text{ and } i \neq j \\ -\log(m'_{i,j}) & \text{if } m'_{j,i} = 0 \text{ and } m'_{i,j} \neq 0 \\ -\log(m'_{j,i}) & \text{if } m'_{i,j} = 0 \text{ and } m'_{j,i} \neq 0 \\ \min\big(-\log(m'_{i,j}), -\log(m'_{j,i})\big) & \text{otherwise,} \end{cases} \tag{6}$$

and

$$d^2_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } m'_{i,j} = m'_{j,i} = 0 \text{ and } i \neq j \\ -\log\big((m'_{i,j} + m'_{j,i})/2\big) & \text{otherwise.} \end{cases} \tag{7}$$

Used passes			d^1 (minimum)		d^2 (average)	
$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{D}_{\min}$	$\mathcal{D}_{\max}$	$\mathcal{D}_{\min}$	$\mathcal{D}_{\max}$
•	○	○	100.00%	100.00%	100.00%	100.00%
•	•	○	174.82%	**241.20%**	204.61%	203.09%
•	○	•	160.95%	216.21%	151.91%	**239.59%**
•	•	•	113.29%	184.68%	126.18%	168.30%

Table 3: Speed-up results for our multi-pass method for the Telephone Database (isolated word recognition, frame-based case), relative to the speed of the basic multi-stack decoding algorithm.

Next let D' be the output of a shortest path-finding algorithm with the input of either the D^1 or D^2 matrix; D' will contain the distance between the phonemes for clustering. So we have four variations depending on the distance function (d^1 and d^2) and the clustering method ($\mathcal{D}_{\min}$ and $\mathcal{D}_{\max}$) applied.

For the clustering stopping criterion L, we plotted the number of clusters for the L value on a diagram and chose L values manually where there was a significant gap between two cluster-unions. This led to several possible choices, all representing a phoneme grouping, thus a possible search pass; these were denoted by $\mathcal{P}_i$-s, $\mathcal{P}_0$ being the original pass. To determine which ones to use and what stack size to employ, exhaustive testing had to be done, performed on two databases (both requiring isolated word recognition): one in a segment- and one in a frame-based context. In the latter fewer potential passes were used due to the lower number of original phonemes (no distinction was made between long and short versions). In a test we varied the used passes and their stack sizes; the quickest configuration was noted which achieved a minimal recognition accuracy (see tables 2 and 3). The speed-up rates indicate that this multi-pass method was quite successful. The many unsuccessful test cases on the Children Database are probably due to the overly restricted phoneme sets $\mathcal{P}_3$ except using d^2 and $\mathcal{D}_{\max}$, but this exception proved to be the fastest among all test cases, and it (the smallest phoneme group $\mathcal{P}_2$ using d^2 and $\mathcal{D}_{\max}$) worked fine on the other database as well.

1.3 Speeding Up Segment-Based Speech Recognition via Segmentation

During search we deal with hypotheses which associate phonemes with segments; these A_i segments can be referenced by their boundary positions $[t_{i-1}, t_i]$. Up till now we set no restrictions on these t_i-s except the trivial ones (they are integer values in increasing order, $1 \leq t_i \leq t$). We may, however, define a set T and allow only its elements; this T is called the set of possible segmentations, and the method which computes T is called the *segmentation algorithm*. Indeed, it is very unlikely that every value between 1 and t will be needed, but keeping unused values in T makes the search process more time-consuming and without any gain. Also, this T in the ideal case coincides with the real boundary between phonemes, which can hopefully be predicted via signal processing techniques [26].

The basic segmentation algorithm draws a possible bound at every kth frame, which is quite a dull, but satisfactorily-working method. Next we present three other methods, which all rely on a b_i bound probability for every a_i frame, supplied by an ANN trained in regression mode on the classic 13 MFCC Δ coefficients. One may think that supplying the b_i values solves the whole segmentation problem; pursuing this idea, the first two methods are straightforward ones. The **Thresholding Algorithm** still places a possible boundary at every k^{th} frame, but only if the corresponding b_i value exceeds a threshold $p_{\min}$ (see Figure 1). This method yielded poor results: parameter adjustments led to a

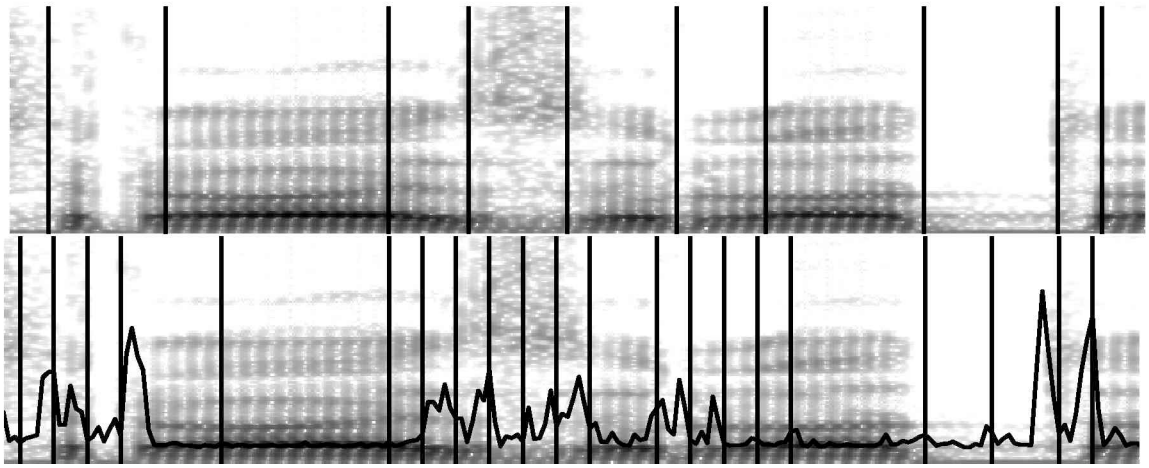

Figure 1: Above: the spectrum of a speech excerpt, along with its correct segmentation. Below: the same spectrum with the b_i values and the output of the Thresholding Algorithm.

Method	Parameters	Correctness	Accuracy	Speed-up rate
Thresholding	$k = 6$, $p = 0.065$	96.06%	94.94%	110.83%
Thresholding	$k = 6$, $p = 0.070$	97.49%	95.70%	87.70%
Maximum	$k = 3$	93.90%	90.68%	78.29%
Maximum	$k = 4$	95.34%	92.83%	137.02%
LIF	$k = 6$, $p = 1.75$	97.85%	96.42%	155.57%
LIF	$k = 7$, $p = 1.30$	98.20%	95.69%	127.26%
Baseline	—	96.42%	95.34%	100.00%

Table 4: Some results obtained using the Thresholding, Maximum and LIF algorithms on the Medical Database (sentence recognition, segment-based case), with their parameters.

small speed-up, but the recognition figures became worse, while other configurations improved the accuracy, but caused some slow-down. These results are probably due to the fact that the change of spectrum (reflected by the b_i values) often occurs over a long time with slightly higher b_i values, which this method could not detect. Our second method, the **Maximum Algorithm** looks for positions where the b_i values take their local maxima over a given interval. By adjusting the size k of the neighbourhood examined, we can set a minimal distance between the suggested boundaries (see Figure 2). Although this method seemed to be the most promising one, in practice it was not so: it inserted few needless bounds, but it also skipped some vital ones, which is a serious mistake. Furthermore, the algorithm was only limitedly tuneable with its single parameter being a small integer.

The failure of these two methods suggests that a more complicated solution is needed. The **LIF Algorithm** seeks to dynamically adjust the density of possible bounds so that it is proportional to the local b_i values. A solution for this is to apply a leaky integrate-and-fire (LIF) neuron model [6]. Its operation is simple: the neuron integrates its input until the sum reaches a certain threshold; then the neuron fires, its potential is reset, and the process starts again. It can be refined by making the integration "leaky", so that the sum decays with a time constant. The model may also incorporate a refractory period, above which it is not excitable. In our segmentation algorithm the b_i values are the input for the integrate-and-fire neuron: where these values are higher, their sum reaches the activation threshold sooner, so the spikes become more dense. The density of the spikes can be controlled by the

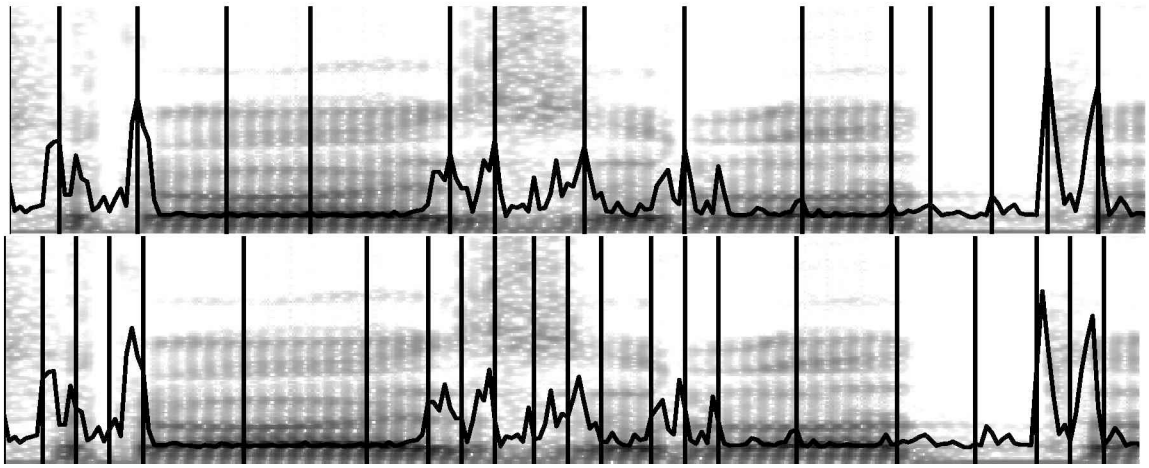

Figure 2: Above: the same spectrum as in Figure 1 with the b_i values and the output of the Maximum Algorithm. Below: the same spectrum with the b_i values and the output of the LIF Algorithm.

threshold p, while by setting the refractory period k a minimum distance between the neighbouring spikes can be guaranteed (see Figure 2). This algorithm, unlike the others, achieved a satisfactory improvement in speed with no loss in accuracy; it was even able to raise the overall recognition scores and get a nice gain in speed (see Table 4). While the Maximum Algorithm was very good at finding possible phoneme bounds, it could not be tuned. The Thresholding Algorithm could be fine-tuned with its two parameters, but the method itself was too primitive. The LIF algorithm, however, seems to be the golden mean between the two: it has two parameters for tuning, suggests phoneme bounds at smarter places than the Thresholding Algorithm does, and it takes the context into account.

Summary of Theses

I/1. The author created a multi-pass search technique to speed up the search process in speech recognition. This method was based on the use of multiple, hierarchical phoneme sets, which were created by clustering the elements of the original phoneme set based on the confusion matrix of phoneme classification. Using the hierarchical property of the phoneme groups, the corresponding passes can be readily implemented as a more compact phoneme group unites several. This way the search process was significantly speeded up (published in [13; 14; 25]).

I/2. The author constructed a method for detecting the interchange of phonemes to more precisely model the bound between them. Based on it, he constructed a novel algorithm to supply a set of possible phoneme boundaries. As shown by the results, this algorithm can indeed lead to a significant speed-up over that of the standard segmentation method, which was our aim. Moreover, a careful configuration of this method also gave a noticeable improvement in the recognition accuracy along with a significant speed-up (published in [18]).

I/3. The author examined the behaviour of the multi-stack decoding algorithm and the Viterbi beam search method. These algorithms both store a number of hypotheses for each possible bound between phonemes. By introducing a number of techniques to more cleverly adjust the number of hypotheses stored, the author was able to speed up the search process along with little or no loss in the recognition accuracy (published in [11; 12; 14; 16; 25]).

2 Improving the Accuracy of Speech Recognition

In the second part we seek to make the speech recognition process more accurate: we inspect areas which could lead to an improvement in the recognition accuracy. As there are a large number of areas which might be able to do this, we had to restrict our scope to just a few such areas. We will concentrate on the application of different aggregation operators in hypothesis cost calculation and on the so-called anti-phoneme problem, as these have practically no additional time requirements.

2.1 Using Aggregation Operators to Calculate Hypothesis Probabilities

Recall that in the discriminative approach we want to find the most probable word $\hat{w} \in W$ for which $P(A|w)P(w)$ is maximal. Concentrating on the former factor we will decompose it into

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

where the word w equals its phoneme sequence $o_1, \ldots, o_j$, and $A = A_1, \ldots, A_n$ is the decomposition of the speech signal $A = a_1, \ldots, a_t$ for non-interleaving segments, one for each phoneme. This formula is based on the assumption of independence; but it was shown [31] that this assumption is actually false, and the use of other operators is also viable as they may fit the actual problem better than the default multiplication. This allows us to use a more general representation [29], namely

$$P(A|w) = g_2\big(P(A_1|o_1), P(A_2|o_2), \ldots, P(A_n|o_n)\big), \tag{9}$$

where g_2 is an operator used to construct word-level probabilities from phoneme-level ones. Following this line, another operator (g_1) is used to supply the phoneme-level $P(A_j|o_j)$ values from local information sources. In a typical **frame-based** environment (as in the case for a Hidden Markov Model (HMM)), g_1 is usually based on the a_j frames and is defined as

$$P(A_j|o_j) = g_1(A_j, o_j) = g_1(a_{t_{j-1}}, \ldots, a_{t_j}, o_j) = \prod_{l=t_{j-1}}^{t_j} c_{o_j}^{t_j - t_{j-1}} \cdot P(a_l|o_j), \tag{10}$$

where c_{o_j} is the state self-transition probability for the state belonging to phoneme o_j, and typically a Gaussian Mixture Model (GMM) supplies the $P(a_l|o_j)$ values. In a **segment-based** case g_1 is usually the length-normalized output of the phoneme classifier [29]. In the following we will change either the g_1 or g_2 operator. This way the probability of word-segment pairs is affected, which can also change the most probable one, preferably improving the accuracy, which was our aim.

2.1.1 Using Mean Aggregation Operators

First we opted for an operator whose behaviour could be tuned between different extremes. It led us to the *mean aggregation operators* taken from fuzzy logic [23], namely the *root-power mean operator* [4]

$$G_\alpha(x_1, \ldots, x_j) = \left(\frac{x_1^\alpha + \ldots + x_j^\alpha}{j}\right)^{\frac{1}{\alpha}}, \qquad \alpha \in \mathbb{R}. \tag{11}$$

It is well-known [20] that when $\alpha \to -\infty$, $G_\alpha \to \min(x_1, \ldots, x_j)$; G_{-1} equals the harmonic mean;

Operator used	α	Accuracy
Nothing (baseline)	–	77.83%
G_α	0.6	84.35%
$G_{\alpha,\lambda}$ with $\lambda = 0.7$	0.4	83.04%
$G_{\alpha,\lambda}$ with $\lambda = 0.8$	0.5	83.91%
$G_{\alpha,\lambda}$ with $\lambda = 0.9$	0.5	84.78%
$G_{\alpha,\lambda}$ with $\lambda = 1.0$	0.6	84.35%

Table 5: The best accuracies for the root-power mean operator and for the decaying root-power mean operator on the Children Database (isolated word recognition, segment-based case).

Operator used	α as g_1	α as g_2	Accuracy
Nothing (baseline)	–	–	94.20%
G_α as g_1	0.94	–	95.13%
G_α as g_2	0.94	1.00	95.13%

Table 6: The best accuracies for the root-power mean operator on the Telephone Database (isolated word recognition, frame-based case). First the α for g_1 was optimized, then we set the α for g_2 too.

when $\alpha \to 0$, G_α tends to the geometrical mean; G_1 equals the arithmetical mean; and when $\alpha \to \infty$, $G_\alpha \to \max(x_1, \ldots, x_j)$. By varying the α parameter we can make a continuous transition from minimum to maximum, which means that this operator is quite flexible, coming in handy in any application. Now we define a variant of this operator, the *decaying root-power mean operator*, by

$$G_{\alpha,\lambda}(x_1, \ldots, x_j) = \left(\frac{\lambda^{j-1} x_1^\alpha + \lambda^{j-2} x_2^\alpha + \ldots + \lambda x_{j-1}^\alpha + x_j^\alpha}{j} \right)^{\frac{1}{\alpha}}, \tag{12}$$

where $\alpha \in \mathbb{R}$ as before and $\lambda \in [0, 1]$ is a weighting parameter used to make the last value more important than the ones before it. In our case the last value refers to the last phoneme or frame of the hypothesis, so this operator seems to be a good variant for our particular problem.

Next we seek to utilize these operators either as g_1 or g_2, which requires setting their parameters. The root-power mean operator can be optimized quite easily with its single parameter: first an interval was determined by preliminary tests where it worked adequately, then this interval was explored with a small step size. For the decaying root-power mean operator, however, we chose a number of λ values, and then the remaining parameter (α) could be set in the same way as before.

In the segment-based case, of course, only g_2 can be changed as here g_1 is not a true operator. Table 5 shows the best accuracies and the corresponding α parameters for the root-power mean operator G_α and for the decaying root-power mean operator $G_{\alpha,\lambda}$ for different λ values. Utilizing G_α led to a big improvement in the recognition accuracy, indicating that perhaps it models the problem better than the default multiplication; but later, the decaying root-power operator achieved only a slight improvement, which suggests that here it has no real advantages over G_α and it is also a more complex one. In the frame-based case, however, it was possible to modify both operators. For the sake of easier usage we did not set their parameters in parallel, but first optimized g_1, leaving g_2 in its basic, multiplication form; then we turned to g_2, keeping g_1 as the optimized operator (see Table 6). We found that setting g_1 led to a significant improvement, but changing g_2 led to no further gain.

t-norm family	T_P	T_L	T_λ^{SS}	T_λ^{H}	T_λ^{Y}	T_λ^{D}	T_λ^{AA}
Best accuracy	92.17%	0.26%	93.47%	92.60%	89.45%	93.47%	93.04%
Rel. error reduction	0.00%	—	16.60%	5.49%	—	16.60%	11.11%

Table 7: Recognition percentage scores and relative error reduction rates for the standard triangular norms applied on the Children Database (isolated word recognition, segment-based case).

t-norm family	T_P	T_L	T_λ^{SS}	T_λ^{H}	T_λ^{Y}	T_λ^{D}	T_λ^{AA}
Best accuracy	91.69%	0.69%	92.61%	92.50%	—	92.25%	92.03%
Relative error reduction	0.00%	—	11.07%	9.74%	—	6.73%	9.74%
Best correctness	92.03%	0.81%	93.31%	93.19%	—	93.03%	92.73%
Relative error reduction	0.00%	—	16.06%	14.55%	—	12.54%	8.78%

Table 8: Accuracy and correctness values and relative error reduction rates with the standard triangular norms applied on the Medical Database (sentence recognition, segment-based case).

2.1.2 Using Triangular Norms

Practically any kind of operator can be used either as g_1 or as g_2, but only a small fraction of these operators could work well. Moreover, we should use operators which can be easily applied and fine-tuned, so we set two criteria: their behaviour should be easily modifiable (having one or more parameters), and, as the default operator is the product one, they should be "multiplication-like". The *triangular norms* are standard operators of fuzzy logic [4; 23], and they fulfil both requirements, thus we applied them. Another reason for using t-norms is that in fuzzy logic they represent AND-like relations, and here we also have AND-connections (i.e. segment A_1 is phoneme o_1 with x_1 probability AND segment A_2 is phoneme o_2 with x_2 probability etc.).

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

(T1)	$T(1,x) = x$	*for all* $x \in [0,1]$,	*(boundary condition)*
(T2)	$T(x,y) = T(y,x)$	*for all* $x, y \in [0,1]$,	*(commutativity)*
(T3)	$T(x,y) \le T(u,v)$	*for any* $0 \le x \le u \le 1$, $0 \le y \le v \le 1$,	*(T is nondecreasing in both arguments)*
(T4)	$T(x,T(y,z)) = T(T(x,y),z)$	*for all* $x, y, z \in [0,1]$.	*(associativity)*

It is easy to see that the product operator is also a t-norm, so we will refer to it as T_P. And

Definition 2 *A t-norm T is said to be*

continuous	*if T as a function is continuous on $[0,1]$;*
Archimedean	*if $T(x,x) < x$ for all $x \in (0,1)$.*

As these two properties seem to be important for a function used either as g_1 or g_2, we will use triangular norms which fulfil both these requirements. A number of well-known triangular norms described by Klement, Mesiar and Pap [23] were tested. A triangular norm is a two-operand function, and (either as g_1 or g_2) we need an operator which can handle any number of arguments, but it can be easily resolved by the associativity property ((T4)). All the norms tested have one parameter (except the Lukasiewicz norm T_L, which has none), giving them the required flexibility, but still remaining

easy to set (which was done in the same way as that for the root-power mean operator). The standard triangular norms were tested in a segment-based context only, applied as g_2, but both in an isolated word recognition task (see Table 7) and in a sentence recognition one (see Table 8). We could achieve a nice reduction in the error rates, depending on the actual standard triangular norm applied.

2.1.3 Using a Two-Parameter General Triangular Norm

In the next step we seek to apply something more general than the standard triangular norms. Dombi introduced a generalized family of triangular norms, which can be represented in the following way:

$$T_{GD,\gamma}^{(\alpha)} = \frac{1}{1 + D_\gamma(x)} \qquad \alpha > 0, \tag{13}$$

where

$$D_\gamma(x) = \left(\frac{1}{\gamma} \left(\prod_{i=1}^{n} \left(1 + \gamma \left(\frac{1 - x_i}{x_i} \right)^\alpha \right) - 1 \right) \right)^{1/\alpha} \tag{14}$$

and $\gamma > 0$ [3]. Furthermore, this operator includes many well-known and widely-used triangular norm families (like the Dombi, Hamacher, Einstein and the product operators) as some special cases.

Its application required some adaptation for the setting of parameters, as it has two (α and γ) instead of the standard triangular norm families which had only one (λ). As α and γ have to be set in parallel, the methods used previously cannot be applied. One way could be the determination of an adequately working interval for both parameters by preliminary tests, which intervals could be explored by some step sizes; but this method raises the number of tests needed to the second power.

We followed another line of thinking: by fixing all the possible settings of a speech recognizer configuration except the parameters of the triangular norm, the accuracy can be considered as a function of these two parameters, which is to be maximized. In our case, where only g_1 or g_2 is changed, it is an optimization problem in a two-dimensional space, and was solved by utilizing a gradient-based optimization method: the SNOBFIT package [21]. The experiments were done in a frame-based context, replacing g_1 and keeping g_2 at its multiplication, in a sentence recognition task. We allowed 500 iterations, which is a fraction of the number of tests required by the first strategy. The error rates were greatly cut (see Table 9): a relative error reduction of above 50% was achieved, which is a much better score than what could be achieved by using a standard triangular norm.

2.1.4 A More General Approach: The Logarithmic Generator Function

Now we shall present a more general way of modelling triangular norms, and supply an application-oriented technique to fit it to the given problem. For this we first have to state another definition.

Theorem 1 *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*

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

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

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

The n-ary extension of this formula is trivially

$$T^N(p_1, p_2, \ldots, p_N) = f^{-1}\Big(\sum_i f(p_i)\Big). \tag{15}$$

This representation is unique up to a positive multiplicative constant. If the t-norm is strictly monotonously increasing, then f is strictly monotonously decreasing, and the pseudo-inverse function $f^{(-1)}$ is the normal inverse f^{-1}. We will apply this case; then f is an *additive generator* of T [4].

Now we can turn to its application. In a typical speech recognition environment, to avoid underflowing, instead of a probability p we use the cost $c = -\log p$. Since it is desirable not to convert cost to probability and vice versa all the time, we first incorporate this conversion into Eq. (15), i.e.

$$-\log\Big(f^{-1}\Big(\sum_{i=1}^{N} f(e^{-c_i})\Big)\Big). \tag{16}$$

As we always apply this formula, it is straightforward to include the calculation of the negative exponential into f; i.e. we will define the logarithmic generator function as

$$\phi(x) = f(e^{-x}). \tag{17}$$

Now we can incorporate $\phi(x)$ into Eq. (16); that is,

$$\phi^{-1}\Big(\sum_{i=1}^{N}\phi(c_i)\Big) = -\log\Big(f^{-1}\Big(\sum_{i=1}^{N} f(e^{-c_i})\Big)\Big) = -\log T(e^{-c_1}, e^{-c_2}, \ldots, e^{-c_N}), \tag{18}$$

so using the logarithmic generator function $\phi(x)$ in the same way as the additive generator function $f(x)$ will lead to the calculation of the same triangular norm T, only with cost values instead of probabilities, both as arguments and as the result. The logarithmic generator function is $\phi(x) : [0,\infty] \to [0,\infty]$, is strictly increasing, $\phi(0) = 0$, and is unique up to a multiplicative constant for a given T t-norm. To model $\phi(x)$ practically any approximation could be used; our solution assumes that it is piecewise linear. Let $\phi = \phi_{a_1,\ldots,a_n}^{m_1,\ldots,m_n} : [0,\infty] \to [0,\infty]$ be a piecewise linear, strictly increasing function with break points on the domain as $0 = a_0 < a_1 < a_2, \ldots, a_n < a_{n+1} = \infty$ and with steepness values $m_1, m_2, \ldots, m_n > 0$ respectively and $m_{n+1} = \lim_{x\to\infty}\phi'(x) = 1$. That is,

$$\phi(x) = (x - a_j)m_{j+1} + \sum_{i=1}^{j}(a_j - a_{j-1})m_j,\ a_j \le x < a_{j+1}. \tag{19}$$

The function ϕ is unique up to a positive multiplicative constant; by setting m_{n+1} to 1, we fix one of these equivalent representations. Furthermore, we can position these control point a_j-s at values where they represent our problem as accurately as possible; so, for given control points, a function ϕ can be described by the vector of the n steepness values ($\overline{m} = (m_1, m_2, \ldots, m_n)$).

The only remaining issue is the choice of control points, for which we suggest a statistic of use: we noted that, during a typical recognition, which cost values had to be handled. The resulting histogram was divided into $n+1$ equal parts, the boundaries being the a_i values (see Figure 3). This way, fixing the value of n, setting the control points, and optimizing $\overline{m}$ for maximum accuracy, we can easily model the logarithmic generator function and so the triangular norm itself. Table 9 shows the performance of this method in a frame-based context (so g_1 was changed while g_2 remained multiplication): using the logarithmic generator resulted in the best performance, but only if n was big enough to give the norm the required flexibility. And, as the logarithmic generator works directly on the cost values, it was the easiest one to apply.

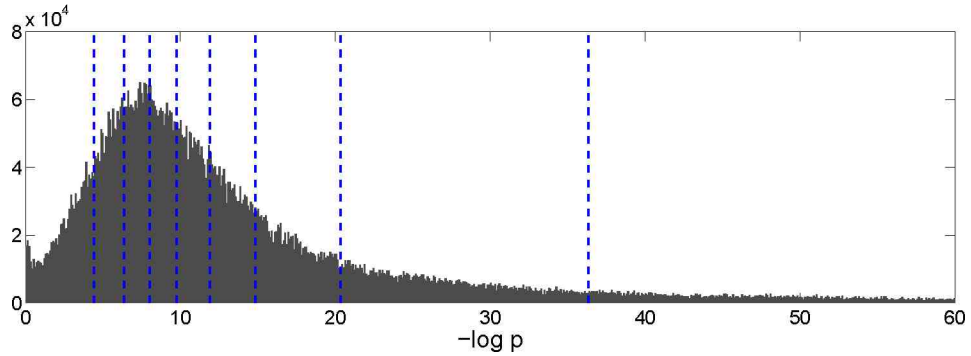

Figure 3: A histogram of the $-\log p$ values appearing during a standard speech recognition process using multiplication, in the interval $[0, 60]$, with the suggested positioning of $n = 8$ control points.

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%
t-norm via logarithmic generator, $n = 8$	98.27%	98.84%	94.00%
t-norm via logarithmic generator, $n = 16$	98.84%	99.19%	96.00%

Table 9: The best accuracy and correctness values obtained for the triangular norms tested on the Medical Database (sentence recognition task), frame-based case, and the ratio of correct sentences.

2.2 The Anti-Phoneme Problem

Finally we will focus on the anti-phoneme problem, arising in a segment-based context only. Its purpose is to decide whether speech segments correspond to a "real" phoneme or not, for which we have a large number of examples for correct segments in the form of a hand-labelled corpus, but there is no sure way of having counter-examples since these should contain anything else occurring in a sound recording (noise, segments longer or shorter than one phoneme, etc.) called "anti-phonemes" [7]. As we only have examples of the positive class, it is an example of *one-class classification* [28].

Recall that in the speech recognition problem we look for the most probable word $\hat{w} \in W$

$$\hat{w} = \arg\max_{w \in W} P(A|w)P(w). \tag{20}$$

Concentrating on $P(A|w)$, we assume that the A signal can be divided into non-overlapping segments. Since this correct segmentation of the signal is not known, it appears as a hidden variable S:

$$P(A|w) = \arg\max_{s \in S} P(A, s|w). \tag{21}$$

There are several ways of decomposing $P(A, s|w)$ further, depending on our modelling assumptions, as S is not well-defined at this point. As the segments are assumed to be independent, the corresponding local probability values will simply be multiplied. For example, in our system we apply [30]

$$\prod_{j=1}^{n} \frac{P(o_j|A_j)P(\overline{\alpha}|A_j)}{P(o_j)}, \tag{22}$$

where the anti-phoneme is denoted by α, thus $P(\overline{\alpha}|A_j)$ denotes the probability that the given segment is *not* an anti-phoneme. Next we will focus on methods for estimating the $P(\overline{\alpha}|A_j)$ probability value to improve the recognition accuracy.

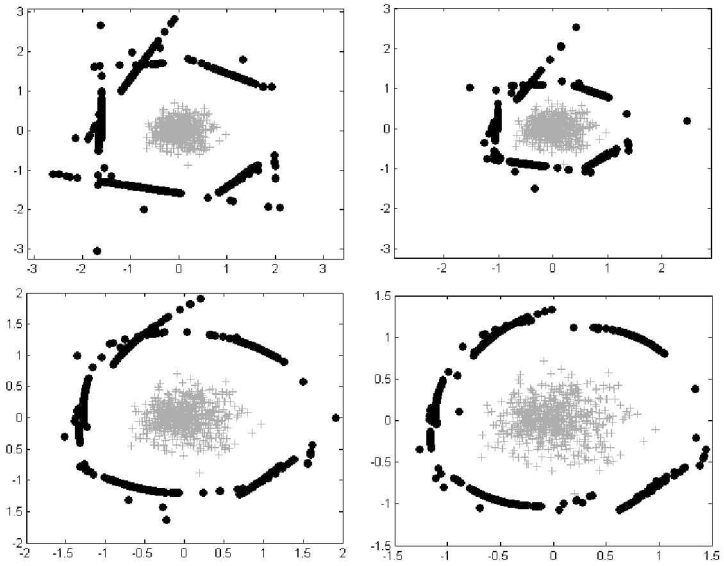

Figure 4: Some generated counter-examples with different settings. Left to right: 1st: $(dist, curv) = (1, 0)$, 2nd: $(dist, curv) = (0.4, 0)$, 3rd: $(dist, curv) = (1, 0.5)$, 4th: $(dist, curv) = (1, 1)$. $curv = 0$ generates points of a hyperplane, while $curv > 0$ generates points of a better fitting hyper-surface; $dist$ controls the distance between the boundary points and the generated counter-examples.

There are two main approaches for solving a one-class classification problem like this. First, we could use a method which can by itself model all the positive examples. The second approach is to take the actual occurrences of phonemes as positive examples, and generate anti-phonemes as negative ones; then the two classes can be separated by some statistical method like ANNs. Of course this automatic generation is not straightforward, and it can also be done in two ways: task-specifically, or by using some general algorithm.

Perhaps the most straightforward way of describing all the phonemes is to model them with the sum of some Gaussian curves. And this is essentially what Gaussian Mixture Models (GMMs) [5] do, hence their application is the default solution in the literature [7]. On the other hand, Tóth introduced a method for generating "incorrect" segments [30]. This algorithm, called "Incorrect Segment Sampling", is based on the idea that the negative examples are probably parts of speech which start and/or end at positions where there is no real bound between phonemes. If we know the real phonetic boundaries – which is the case for any training database –, then we can generate negative examples by choosing incorrect phoneme boundaries for the start and/or end segment bound. The actual solution is quite straightforward: for each phoneme several anti-phonemes are generated by placing one or both phoneme boundaries earlier or later by δ milliseconds.

Besides applying a speech recognition-specific method, we can also use a general counter-example generation algorithm, with the set of positive examples X as its input and a set of negative or counter-examples as its output. Bánhalmi introduced one such counter-example generation algorithm [2], which we applied for the anti-phoneme task as the third method. The main idea behind this algorithm is to project each positive example point $x \in X$ outside X. To do this, first the set B of the boundary elements of X is calculated, then each positive example is transformed over the closest boundary point with the use of two parameters: $curv$ controls the curvature of the resulting surface, while $dist$ sets the distance of the resulting point from the appropriate boundary point. Finally the resulting point

Method	Accuracy	Correctness
GMM with 20 Gaussians	91.12%	91.80%
Incorrect Segment Sampling	91.97%	92.62%
General C.E., $curv = 0.0$, $dist = 1.5$	95.58%	95.82%
No anti-phoneme modelling	88.17%	88.53%

Table 10: Some notable test results for all three methods on the Medical Database (sentence recognition, segment-based case).

is checked to see whether it is an outer point of the original positive examples, which is done in the same way as the boundary points were identified. If it is not a real outer point, then the process has to be repeated with the second-closest boundary point, etc. This is done for each element of X, resulting in N negative examples for N positive ones (see Figure 4).

This method has an obvious weakness: its time requirement for larger databases is very high. As in our case there were indeed large datasets, we had to deal with this issue. It meant that executing this algorithm with different $dist$ and $curv$ parameters took rather a long time. To solve this problem – which was essential for the practical use of this algorithm – we introduced a number of modifications.

- We noticed that this algorithm could be divided into two distinct parts: the first one calculates the boundary points, while the second one is responsible for the actual counter-example generation. Fortunately the first one is responsible for the higher time complexity; the $dist$ and $curv$ parameters, however, appear in the second part only. Thus, no matter how many parameter combinations we seek to test, the first – and much slower – part has to be computed just once.

- This second part could be divided up still further: when we know the elements of the boundary set, for each element x first we have to find the closest boundary point $x_b \in B$. This procedure also has nothing to do with the varying parameters, so it can be separated from the actual projection and should also be computed just once (although strictly after determining the boundary elements), then its result can be stored. Thus, when trying out a new $dist$ and $curv$ parameter value pair, we only have to read the index of the closest boundary instead of calculating it again as was done in the original version of this algorithm.

- Next we found that the last part where we test whether the resulting point is an outer point is also not necessarily needed: for larger $dist$ values (in our case $dist \geq 1.5$) the projected point lies too far away from the original example set to be an inner point. Of course the smaller the $dist$ value is, the more likely that some of the projected points may be inner points, but this risk is acceptable because omitting this part leads to an overwhelming speed-up.

Using these modifications it became possible to test all three methods on a sentence recognition task (see Table 10); the features were the same as those used for phoneme classification. For the baseline value we did not use any anti-phoneme model at all. The general counter-example generator method proved to be the most successful one, but it could not have been applied without the modifications introduced, as it would have been practically impossible to set its two parameters satisfactorily due to the CPU time requirements.

	[13]	[18]	[16]	[11]	[12]	[14]	[25]	[15]	[24]	[10]	[17]	[9]
I/1.	•					•	•					
I/2.		•										
I/3.			•	•	•	•	•					
II/1.					•	•	•	•	•	•		
II/2.											•	
II/3.												•

Table 11: The relation between the theses and the corresponding publications

Summary of Theses

II/1. In the search task of speech recognition the generated hypotheses are ranked by their *cost values*, usually calculated in two steps: first *phoneme-level* costs are aggregated from the *frame-level* ones, then the cost of the hypothesis is calculated from the phoneme-level values. The author applied a number of fuzzy functions at these levels: *mean aggregation operators* and *triangular norms* were tested to improve the recognition accuracy. He also modified the root-power mean operator by introducing a weighting parameter to make earlier observations less important. Furthermore, to set the two parameters of a generalized triangular norm, he turned the problem into a minimization one in a two-dimensional space and applied a global optimization method. The improvements in accuracy indicate that these operators fit the problem better than default multiplication (published in [10; 12; 14; 15; 24; 25]).

II/2. The author proposed an application-oriented method for modelling triangular norms by introducing the logarithmic generator function. Assuming that this function was piecewise linear, he introduced a technique to adequately fit it to the actual application based on the histogram of the occurring probabilities. He applied this method in the speech recognition minimization problem introduced in Thesis II/1, and showed that it can improve the performance by a greater amount than any other classical norms tested (published in [17]).

II/3. The author applied a general counter-example generator method from the literature in the anti-phoneme problem of segment-based speech recognition. To make it possible to use he also introduced several modifications to this method, which were essential to reduce its time requirement. Doing this, he was able to markedly reduce the error rates of the speech recognition system, even when compared to other standard solutions for this problem (published in [9]).

Conclusions

In the author's opinion the most important aspects of speech recognition are accuracy and speed, as we have to recognize the speech signal uttered as accurately as possible, and possibly within real time. He sought to introduce improvements for both, and it resulted in a number of techniques which, based on the results of the experiments, proved to be successful. Most methods have parameters which, of course, probably have to be set again in a different environment, but the test results described here confirm the potential benefits of these methods. Some experiments led to some quite general techniques like the logarithmic generator function, and the refinements over the general counter-example generator algorithm, which could be applied in areas outside of speech recognition as well.

References

[1] L. R. Bahl, P. Gopalakrishnan, and R. L. Mercer. Search issues in large vocabulary speech recognition. In *Proceedings of the 1993 IEEE Workshop on Automatic Speech Recognition*, Snowbird, UT, 1993.

[2] A. Bánhalmi, A. Kocsor, and R. Busa-Fekete. Counter-example generation-based one-class classification. In *Proceedings of ECML*, pages 543–550, 2007.

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

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

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

[6] W. Gerstner and W. M. Kistler. *Spiking Neuron Models*. Cambridge University Press, 2002.

[7] J. R. Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17(2):137–152, 2003.

[8] G. Gordos and G. Takács. *Digital Speech Processing (in Hungarian)*. Műszaki Könyvkiadó, 1983.

[9] G. Gosztolya, A. Bánhalmi, and L. Tóth. Using one-class classification techniques in the anti-phoneme problem. In *Proceedings of the 2009 Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, volume LNCS 5524, pages 433–440, Porto, Portugal, 2009.

[10] G. Gosztolya, J. Dombi, and A. Kocsor. Applying the Generalized Dombi Operator family to the speech recognition task. *Journal of Computing and Information Technology*, 17(3):285–293, 2009.

[11] G. Gosztolya and A. Kocsor. Improving the multi-stack decoding algorithm in a segment-based speech recognizer. In *Proceedings of the 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, volume LNCS 2718, pages 744–749, Loughborough, England, UK, 2003.

[12] 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 (TSD)*, volume LNCS 3206, pages 315–322, Brno, Czech Republic, 2004.

[13] G. Gosztolya and A. Kocsor. A hierarchical evaluation methodology in speech recognition. *Acta Cybernetica*, 17(2):213–224, 2005.

[14] G. Gosztolya and A. Kocsor. Speeding up dynamic search methods in speech recognition. In *Proceedings of the 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, volume LNCS 3533, pages 98–100, Bari, Italy, 2005.

[15] G. Gosztolya and A. Kocsor. 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.

[16] G. Gosztolya, A. Kocsor, L. Tóth, and L. Felföldi. Various robust search methods in a Hungarian speech recognition system. *Acta Cybernetica*, 16(2):229–240, 2003.

[17] G. Gosztolya and L. L. Stachó. Aiming for best fit t-norms in speech recognition. In *Proceedings of the 2008 International Symposium on Intelligent Systems and Informatics (SISY) (IEEE)*, pages 1–5, Subotica, Serbia, Sept 2008.

[18] G. Gosztolya and L. Tóth. Detection of phoneme boundaries using spiking neurons. In *Proceedings of the 2008 International Conference on Artificial Intelligence and Soft Computing (ICAISC)*, volume LNCS 5097, pages 782–793, Zakopane, Poland, 2008.

[19] D. J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[20] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1968.

[21] W. Huyer and A. Neumaier. Snobfit – stable noisy optimization by branch and fit. *ACM Transactions on Mathematical Software*, 35(2):1–25, 2008.

[22] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1997.

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

[24] 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.

[25] 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):95–107, 2006.

[26] T. N. Sainath. Acoustic landmark detection and segmentation using the McAulay-Quatieri sinusoidal model. Master's thesis, MIT, 2005.

[27] R. Schwartz, L. Nguyen, and J. Makhoul. *Multiple-pass Search Strategies*, chapter 18, pages 429–456. Kluwer Academic Publisher, Philadelphia, PA, 1996.

[28] D. M. Tax. *One-class classification; Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology, 2001.

[29] L. Tóth, A. Kocsor, and J. Csirik. On Naive Bayes in speech recognition. *International Journal of Applied Mathematics and Computer Science*, 15(2):287–294, 2005.

[30] L. Tóth, A. Kocsor, and G. Gosztolya. Telephone speech recognition via the combination of knowledge sources in a segmental speech model. *Acta Cybernetica*, 16(4):643–657, 2004.

[31] S. Young. Statistical modelling in continuous speech recognition. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, Seattle, 2001.