4th International Conference on Applied Informatics
Eger–Noszvaj, Hungary, August 30–September 3, 1999.

An overview of the OASIS speech recognition project

András Kocsor, Kornél Kovács, András Kuba Jr., László Tóth

Research Group on Artificial Intelligence, József Attila University
e-mail: kocsor@inf.u-szeged.hu
e-mail: coree@rgai00.inf.u-szeged.hu
e-mail: kandras@rgai00.inf.u-szeged.hu
e-mail: tothl@inf.u-szeged.hu

Abstract

This paper presents an overview of the "OASIS" segment-based speech recognition system developed at the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences. We present the preprocessing method, the features extracted from its output, and how segmentation of the input signal is done based on those features. We also describe the two types of evaluation functions we applied for phoneme recognition, namely a C4.5 and an instance-based learning technique. In our system, the recognition of words from a vocabulary means a special search in a hypothesis space; we present how this search space is handled and the search is performed.

Our results demonstrate that for small vocabularies we obtained acceptable recognition rates of about 90% even with the very few features and small training database used. It is now a matter of further investigation to see how much these methods could be extended to be applicable to large vocabulary speech recognition.

Key Words: speech recognition, acoustic features, C4.5

1 Introduction

After early attempts in the 70's to build knowledge-based speech recognition systems engineers in the 80's turned to much simpler, mathematically tractable (numerical) statistical learning techniques. These systems, built on Hidden Markov Models (HMM) and Artificial Neural Nets (ANN), dominate both the market and researches since then. At the end of the 90's, automatic speech recognition (ASR) finally reached the level of practical usability. However, in the last ten years there have been only a few improvements in the underlying technology, and the sceptics say this success is rather due to the increase in processor power and the amount of training speech corpora available. Many experts say that these numerical models have reached a limit, and further radical improvements are possible only if the underlying model is changed [1] in order to incorporate a lot of new information we concieved about human speech perception.This means, more knowledge has to be integrated into the recognition technique in a way that keeps the possibility of extracting additional knowledge, with the aid of learning methods.

In our experimental speech recognition system "OASIS" (which stands for "Our Acoustics-based Speaker-Independent Speech recognizer") we try both to bring back AI into speech recognition, and incorporate new knowledge about human speech perception. There are only a few similar systems we know, the closest being the SUMMIT system of MIT [2].

Since building a whole ASR system is an enormous task, we chose a relatively easy first goal: to recognize Hungarian numbers such as "two-hundred and sixty five". This leads to a continuous speech recognition task over a dictionary of 26 words. Both the training and testing database was recorded in office quality (at a sampling rate of 22050Hz), containing carefully pronounced (read-out) speech. Our first results show that at this quality and with such a small dictionary even with quite few and simple features and a little training the system can reach an acceptable recognition rate. It is now a matter of further research to decide whether this approach could handle bad quality spontaneous speech with a large dictionary.

Recognition starts with a preprocessing phase in which the spectral representation of the signal is calculated and acoustic features are extracted from the spectrogram. We especially kept in mind those new results which claim that humans process frequency channels independently, and integration occurs only at a higher stage [3]. Thus our main features were the energies in reasonably chosen frequency bands.

Following this we get a rudimentary segmentation. Information over segments are integrated, in the form of what we call "interval-features". Since the segmentation is not reliable (it is supposed to give an over-segmentation), possibilities of fusing these segments have to be examined. This leads to a search problem which is solved by a depth-first search over the possible strings provided by a dictionary. We use punishment-values to distinguish among possible solutions and the result of the recognition is the string that has the smallest accumulated punishment.

2 Preprocessing

In the preprocessing phase the system converts the speech signal into the traditional (wide-band) spectrogram, that is, calculates its short-time Fourier spectrum (using a Hamming window). The intensity of the spectrum is, as traditional, taken on a logarithmic (decibel) scale. Occasional smoothing, Bark-warping and other similar transformations take place in the feature extraction phase, where necessary.

For the sake of clarity we shall briefly formalize the preprocessing step. For simplicity we use continuous notation, but in the practice both the signal and its spectrum are given by their samples, of course. In our system we calculate 512 samples from the spectrum between 0 and 11025Hz in every 10ms.

2.0.1 Notations

Let us denote the moments of time by T_{pnt} where $T_{pnt} := I\!R^+$ the non-negative real numbers. We introduce the notation T_{intv} for time intervals: $T_{intv} := \{(t_1, t_2) \in T_{pnt} \times T_{pnt} : t_1 < t_2\}$. The input speech is given by $v : T_{pnt} \to I\!R$ which represents the signal as a function of time. After decomposition of the signal into sinusoid waves the valid range of frequencies is denoted by H. (Usually, $H = [0, 11025]$)

Definition 2.1 *We define the result of preprocessing as an* $Spc : T_{pnt} \times H \to I\!R^+$ *function for which:*

$$Spc(t,h) := \left| \int_{-\infty}^{\infty} v(l)w(l-t)e^{-ihl}dl \right|$$

holds. This function determines the intensity of frequency h at the moment t. w is the Hamming window employed by the system.

3 Acoustic features

This phase is absent in standard speech recognizers; there the values of the smoothed spectrum are considered as "features". In our system we made, and plan to make more experiments with extraction of phonetically meaningful features like "sonorancy", "voicedness" and such [5], but currently the energies of four spectral bands serve as our main features, which is a very coarse representation, but surprisingly it yielded quite good results. We use two types of features in the system; the ones belonging in the first type are called "time-point features", which means they are defined at each time point of the signal and simulate the output of feature extractor neurons. After this, a coarse segmentation of the signal is performed (also based on certain features). It is supposed that in later stages the brain integrates information over these segments; this is simulated by our "interval-features", or "cues", which are defined in a time-interval. The values of these interval-features form the basis of the recognition.

Definition 3.1 *A $f : T_{pnt} \rightarrow I\!R^+$ function is called time-point feature by definition iff $f(t)$ depends only on the intensity of frequencies at the moment t.*

Here we show some examples of time-point features some of which will be of special interest later on:

$$\begin{array}{lcll} f^1_{[a,b]}(t) & := & \int_a^b Spc(t,h)dh, & 0 \leq a < b, \\ f^2_{[a,b]}(t) & := & \max_{a \leq h \leq b} Spc(t,h)dh, & 0 \leq a < b, \\ f^3_{[a,b]}(t) & := & \min_{a \leq h \leq b} Spc(t,h)dh, & 0 \leq a < b. \end{array}$$

Definition 3.2 *A $g : T_{intv} \rightarrow I\!R^+$ function is called interval feature iff $g(t_1,t_2)$ is defined by the values of $Spc(t,h)$, $(t_1 \leq t \leq t_2)$.*

Interval features generated from an arbitrary time-point feature $f(t)$ are:

$$\begin{array}{lcllcl} g^1_f(t_1,t_2) & := & \max_{t\in[t_1,t_2]} f(t) & g^3_f(t_1,t_2) & := & g^1_f(t_1,t_2) - g^2_f(t_1,t_2) \\ g^2_f(t_1,t_2) & := & \min_{t\in[t_1,t_2]} f(t) & g^4_f(t_1,t_2) & := & \frac{\int_{t_1}^{t_2} f(t)dt}{t_2-t_1} \end{array}$$

Features for later use are:

$$\begin{array}{lcllcl} \kappa^1(t_1,t_2) & := & \frac{\int_{t_1}^{t_2} f^1_{[0,800]}(t)dt}{t_2-t_1} & \kappa^2(t_1,t_2) & := & \frac{\int_{t_1}^{t_2} f^1_{[800,1800]}(t)dt}{t_2-t_1} \\ \kappa^3(t_1,t_2) & := & \frac{\int_{t_1}^{t_2} f^1_{[1800,4500]}(t)dt}{t_2-t_1} & \kappa^4(t_1,t_2) & := & \frac{\int_{t_1}^{t_2} f^1_{[4500,11025]}(t)dt}{t_2-t_1} \\ \kappa^5(t_1,t_2) & := & \frac{\int_{t_1}^{t_2} f^1_{[0,11025]}(t)dt}{t_2-t_1} & \kappa^6(t_1,t_2) & := & t_2 - t_1 \end{array}$$

$$\kappa^7(t_1,t_2) := \left(\max_{t\in[t_1,t_2]} f^1_{[0,11025]}(t)\right) - \left(\min_{t\in[t_1,t_2]} f^1_{[0,11025]}(t)\right)$$

During the empirical investigation we used the interval features denoted by κ. All, except from, $\kappa^6(t_1,t_2)$ were derived from $f^1_{[a,b]}$ (defined above). $\kappa^6(t_1,t_2)$ is a trivial interval-feature.

4 Segmentation of speech

Definition 4.1 *An array of* $\mathbf{s}_k = [t_0, t_1, \cdots, t_k]$ *is called segmentation if* $0 = t_0 < t_1 < \cdots < t_k$ *holds.*

A segmentation is called *ideal* if every phoneme in the speech fits onto one $[t_i, t_j]$ interval where $i, j \in \{0, \cdots, k\},\ i < j$. Our aim is to produce an ideal segmentation where $j - i < 6$ holds for every phoneme. This restriction reduces the size of the search space significantly and it should not be a difficult task for any segmenting algorithm.
In our system the segmentation is obtained with the help of the following algorithm:

- We divide the spectra into four part and take one time-point feature for each that characterizes it, namely: $\alpha_1 = f^1_{[0,800]}(t), \alpha_2 = f^1_{[800,1800]}(t), \alpha_3 = f^1_{[1800,4500]}(t)$ and $\alpha_4 = f^1_{[4500,11025]}(t)$.
- Then we construct the function $l_c(t) = \max_{1 \le i \le 4} |\alpha_i(t - c) - \alpha_i(t + c)|$ with an appropriate constant c. In general $c = 20$ millisecond was found satisfactory.
- Then the segment bounds are placed at the local maxima of $l_c(t)$.

This method results in a segmentation that can be regarded as ideal from the point of view of the application.

5 Hypothesis space

The method presented below is suitable for any type of dictionary and any kind of language. However for a concrete practical application we sought to focus on a particular problem. We chose to develop a system that recognizes spoken numbers in the Hungarian language. From now on, *word* means series of phonemes. Phonemes are denoted by p, for instance $p_1 \cdots p_j$ means a word containing j phonemes.

5.1 Dictionary

The dictionary contains the spoken forms of the words we plan to recognise. The words are stored as phoneme series. According to the particular goal we wanted to achieve (i.e. to identify spoken numbers) our dictionary was built from words and word parts that allow us to phonetically describe all the numbers between 0 and 999,999,999 using the concatenation operator. In the Hungarian language this meant 26 different dictionary entries.

5.2 Hypothesis space

Let W be the set of words meaning numbers between 0 and $10^9 - 1$. Let $Pref_k(W)$ mean the set of the k-long prefixes of all the words in W that contain at least k phonemes. For a given $\mathbf{s}_n = [t_0, t_1, \cdots, t_n]$ segmentation which defines n intervals we can make the set $S_k = \{[t_{i_0}, t_{i_1}, \cdots, t_{i_k}] : 0 = i_0 < i_1 < \cdots < i_k \le n\}$ which we call the set of sub-segmentations over $\mathbf{s}_n$ with k elements (intervals). Furthermore, to reduce the number of elements in S_k we can assume that $i_{l+1} - i_l < 6,\ 0 \le l \le k - 1$.

Now we shall recursively define the search tree. Let us denote the root by v_0 and link it to every element of the set $Pref_1(W) \times S_1$. These are the first level vertices.

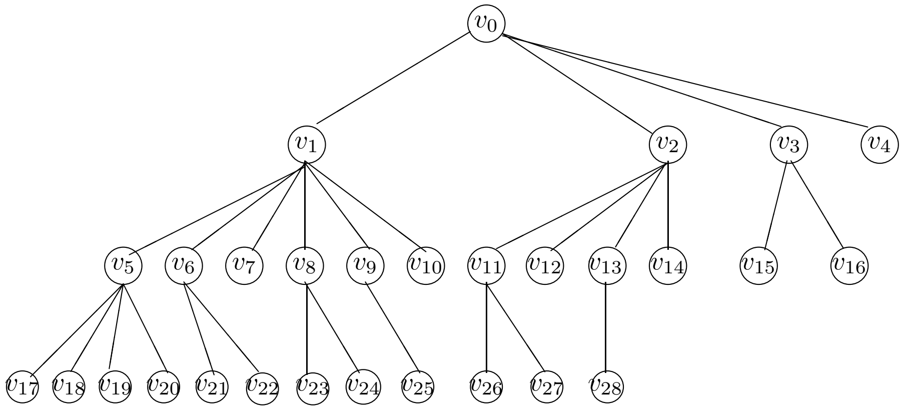

Figure 1: Tree representation of the hypothesis space with $\mathbf{s_4} = [0, 100, 160, 200, 270]$ segmentation and $W = \{p_1p_2p_3, p_1p_2p_4, p_1p_3p_4\}$ as a dictionary

Having a particular $(p_1p_2 \cdots p_j, [t_{i_0}, \cdots, t_{i_j}])$ leaf given, add all

$$(p_1p_2 \cdots p_jp_{j+1}, [t_{i_0}, \cdots, t_{i_j}, t_{i_{j+1}}]) \in Pref_{j+1}(W) \times S_{j+1}$$

points to the tree as descendants of the given leaf. Repeat this step until there are no points to be added. Then the tree is complete. Please note that every vertex in the tree at level n has n phonemes and n interval. Ancestors are similar to their descendants except the last phoneme and the last interval.

During recognition our aim is to reach a leaf $(p_1p_2 \cdots p_j, [t_{i_0}, \cdots, t_{i_j}])$ such that $p_1p_2 \cdots p_j \in W$ holds. We will call this kind of leaves *terminating leaves.*

In Figure 1 we present a hypothesis space with four different phonemes: p_1, p_2, p_3 and p_4. Let us suppose a segmentation $\mathbf{s_4} = [0, 100, 160, 200, 270]$ and a dictionary consisting of the words $p_1p_2p_3, p_1p_2p_4$ and $p_1p_3p_4$. The table below works as a legend for figure 1; it describes the vertices v_i, $(1 \leq i \leq 28)$ on the figure.

v_1	:=	$(p_1,[0,100])$	v_9	:=	$(p_1p_3,[0,100,200])$	v_{17}	:=	$(p_1p_2p_3,[0,100,160,200])$
v_2	:=	$(p_1,[0,160])$	v_{10}	:=	$(p_1p_3,[0,100,270])$	v_{18}	:=	$(p_1p_2p_3,[0,100,160,270])$
v_3	:=	$(p_1,[0,200])$	v_{11}	:=	$(p_1p_2,[0,160,200])$	v_{19}	:=	$(p_1p_2p_4,[0,100,160,200])$
v_4	:=	$(p_1,[0,270])$	v_{12}	:=	$(p_1p_2,[0,160,270])$	v_{20}	:=	$(p_1p_2p_4,[0,100,160,270])$
v_5	:=	$(p_1p_2,[0,100,160])$	v_{13}	:=	$(p_1p_3,[0,160,200])$	v_{21}	:=	$(p_1p_2p_3,[0,100,200,270])$
v_6	:=	$(p_1p_2,[0,100,200])$	v_{14}	:=	$(p_1p_3,[0,160,270])$	v_{22}	:=	$(p_1p_2p_4,[0,100,200,270])$
v_7	:=	$(p_1p_2,[0,100,270])$	v_{15}	:=	$(p_1p_2,[0,200,270])$	v_{23}	:=	$(p_1p_3p_4,[0,100,160,200])$
v_8	:=	$(p_1p_3,[0,100,160])$	v_{16}	:=	$(p_1p_3,[0,200,270])$	v_{24}	:=	$(p_1p_3p_4,[0,100,160,270])$
						v_{25}	:=	$(p_1p_3p_4,[0,100,200,270])$
						v_{26}	:=	$(p_1p_2p_3,[0,160,200,270])$
						v_{27}	:=	$(p_1p_2p_4,[0,160,200,270])$
						v_{28}	:=	$(p_1p_3p_4,[0,160,200,270])$

While $v_4, v_7, v_{10}, v_{12}, v_{14}, \cdots, v_{28}$ are leaves of the tree, only $v_{17}, \cdots, v_{28}$ are terminating leaves.

5.3 Evaluation function

We shall define a δ function that maps a non-negative real value to any arbitrary $[t_1, t_2]$ time interval and p phoneme. The value of δ is lower if p fits well onto the input signal between $[t_1, t_2]$ and is higher if p does not fit. How such a function is obtained is discussed in the next section of the paper.

Assuming we have this δ we can define a weight function named Δ on every node of the search tree as follows: $\Delta(p_1 p_2 \cdots p_j, [t_{i_0}, \cdots, t_{i_j}]) := \sum_{k=1}^{j} \delta(t_{i_{k-1}}, t_{i_k}, p_k)$. Our task will be to search among the terminating leaves and find the one that has a minimal weight (according to Δ).

5.4 Search method

Naturally there are many methods to scan the hypothesis space with. However, two basic ideas are worth considering:

- During the search we should mark the best solution so far.
- If the node under investigation has a greater weight than the best solution presently we can skip over this node and all its descendants. This is due to the monotonicity of Δ.

Our method was a slightly modified version of depth-first search algorithm in which we used the ideas above. At any node of the search tree the algorithm dives to the most promising unvisited child. If there are no more unvisited child nodes the back-track step pops up to the parent node. If a terminating leaf is reached then the value is compared to the best value so far, and optionally stored. During diving and back-track we lock node v as a dead end if $\Delta(v)$ is better than the best result so far.

6 Various evaluation functions

Up to this point we have described a fairly general system. As soon as we define one particular evaluator function, however, it determines the behaviour of the whole application. We will address two essentially different evaluator functions in this section but they have one thing in common, namely they require a database which we use to build them.

6.1 Database

As mentioned above, 26 words are enough to build the Hungarian number names from 0 to $10^9 - 1$ with concatenation. Our group made a small but to some degree representative database from these words. 10 people (males, females and children) were asked to pronounce those 26 words twice, and we recorded them at 22050Hz sampling rate. The created files together then constituted our sample base.

This base went through the pre-processing phase and segmentation was done manually. In this way we obtained a database that has phonemes as the smallest entries. The total number of the phonemes was about 2000, there being 32 different kind of them. Denoting the database with A, it could be described as follows:

$$
\begin{aligned}
A \quad := \{ \quad & [(t_{i_1^1}, t_{i_1^1+1}), (t_{i_2^1}, t_{i_2^1+1}), \cdots, (t_{i_{l_1}^1}, t_{i_{l_1}^1+1}), p_1], \\
& \cdots \\
& [(t_{i_1^{32}}, t_{i_1^{32}+1}), (t_{i_2^{32}}, t_{i_2^{32}+1}), \cdots, (t_{i_{l_{32}}^{32}}, t_{i_{l_{32}}^{32}+1}), p_{32}]\}
\end{aligned}
$$

A has a single line for every p_j, and the (t_i, t_k) intervals are the locations in A where p_j occurs.

6.2 Evaluation functions

First we have to choose r different interval features, namely $\tau^1(t_1, t_2), \cdots, \tau^r(t_1, t_2)$. The more they characterise the phonemes the better they are. In our case which is described in the results section we used $\kappa^1, \cdots, \kappa^7$ as they are defined in Section 3. We have to show how to generate δ from these. With a given δ, Δ is to be computed as mentioned in 5.3.

6.2.1 Statistical averages based weighting function

Let

$$\delta(t_1, t_2, p_j) := \sum_{c=1}^{r} \left(\exp\left(\frac{(\tau^c(t_1, t_2) - o(p_j, c))^2}{\sigma^2(p_j, c)} \right) - 1 \right),$$

where $o(p_j, c)$ is the average of τ^c values for a given p_j phoneme at every occurrence of p_j in the database and $\sigma^2(p_j, c)$ defines the standard deviation of the same values:

$$o(p_j, c) := \frac{\sum_{s=1}^{l_j} \tau^c(t_{i_s^j}, t_{i_s^j+1})}{l_j}, \qquad \sigma^2(p_j, c) := \sum_{s=1}^{l_j} \frac{\left(o(p_j, c) - \tau^c(t_{i_s^j}, t_{i_s^j+1})\right)^2}{l_j}.$$

6.2.2 C4.5 based weighting function

We used a dedicated software package with built-in C4.5 capabilities [4]. The training database was a restricted version of A, one speaker being left out. The output of the C4.5 learning mechanism was a $\hat{T}$ decision tree. For a given (t_1, t_2) interval of the pre-processed speech signal $\hat{T}$ results in one phoneme of the phoneme set according to the values of $\tau^1(t_1, t_2), \cdots, \tau^r(t_1, t_2)$. Let us denote the result phoneme with $\hat{T}(\tau^1(t_1, t_2), \cdots, \tau^r(t_1, t_2))$. As the learning process is not 100 percent accurate we defined a conditional probability matrix (confusion matrix) P with the aid of the database A. A P_{jk} element in the matrix represents the probability of that $\hat{T}$ maps the p_k phonemes in A into p_j. Obviously higher values in the diagonal of P mean better learning results.

By definition:

$$P_{jk} := \frac{\left| \left\{ j : p_j = \hat{T}(\tau^1(t_{i_s^k}, t_{i_s^k+1}), \cdots, \tau^r(t_{i_s^k}, t_{i_s^k+1})),\ 1 \le s \le l_j \right\} \right|}{l_j}, \quad 1 \le j, k \le 32.$$

δ is defined, using the values of P, as:

$$\delta(t_1, t_2, p_j) := 1 - P_{jk}, \text{ where } \hat{T}(\tau^1(t_1, t_2), \cdots, \tau^r(t_1, t_2)) = p_k.$$

7 Results

We should recall that database A contains samples from 10 different speaker. By taking out the samples belonging to one particular speaker, we created $A_1, \cdots, A_{10}$ restricted databases. The databases were segmented manually. For each of these databases we created the statistical average-based evaluator function (SABEF, see 6.2.1) and the C4.5 based evaluator function (see 6.2.2). Then we run a recognizer with every evaluator function on the training database and on the words that were left out, as well. The table below contains the results achieved with the different evaluator functions obtained from $A_1, \cdots, A_5$. The values show the percentage of the correct identification of words using a specific Δ on two test inputs: on one hand the database that was used for obtaining Δ (marked as "TRAINING") and on the other hand the words that were omitted from the training database (marked as "TEST").

SABEF	TEST	TRAINING	C4.5	TEST	TRAINING
Δ_1^1	94.23	92.03	Δ_1^2	96.15	92.58
Δ_2^1	94.23	92.30	Δ_2^2	94.23	92.86
Δ_3^1	92.30	93.13	Δ_3^2	92.30	93.13
Δ_4^1	90.38	92.03	Δ_4^2	90.38	93.40
Δ_5^1	76.92	93.96	Δ_5^2	82.69	94.50
averages	89,61	92.69	averages	91.15	93.29

7.1 Conclusion

Summarizing the results, we can say that the present system produces correct output in 90-92% of cases on both type of inputs (test & training). This is true regardless whether we use C4.5 learning or the average-based functions. Considering the present (slightly artificial) conditions that manifests in relatively small database and few interval features we consider these results satisfactory.

There are some promising results with automatic segmentation as well. However, it is one of our most important tasks to make an exhaustive investigation concerning the efficiency of the segmentation algorithm. We also plan a slight modification in the average based weight function, namely to use weighting factors when cumulating the interval features. The weights are to be defined by the training set so that they could reinforce the characterising power of the interval features. Finally, addition of new interval features would be necessary to increase the discrimination power of the system. Currently we exploit the fact that the dictionary does not contain very similar words, but for such a case introducing special interval features that allow of fine phonetic distinctions would be crucial.

References

[1] H. Bourlard, H. Hermansky, N. Morgan: Towards Increasing Speech Recognition Error Rates, *Speech Communication*, Vol. 18 (1996), 205-231.

[2] J. Glass, J. Chang, M. McCandless: A Probabilistic Framework for Features-Based Speech Recognition, *Proc. Int. Conf. on Spoken Language Processing*, October 1996, Philadelphia, PA, 2277-2280

[3] J.B. Allen: How do Humans Process and Recognize Speech? *IEEE Trans. Speech Audio Process.*, Vol. 2. No. 4, 567-577.

[4] J.R. Quinlan: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.

[5] A.V. Hansen: Acoustic-Phonetic Features used in Automatic Speech Recognition, *Ph.D. dissertation*, Aalborg University, Denmark, 1999.