International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 4 (2012) pp. 358–365
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Speech Recognition by Indexing and Sequencing

Simone Franzini[1] **and Jezekiel Ben-Arie**[2]

[1]Department of Computer Science, University of Illinois at Chicago,
851 S. Morgan St., Chicago, IL 60607, USA
sfranz3@uic.edu

[1]Department of Electrical and Computer Engineering, University of Illinois at Chicago,
851 S. Morgan St., Chicago, IL 60607, USA
benarie@ece.uic.edu

Abstract: **Recognition by Indexing and Sequencing (RISq) is a general-purpose example-based method for classification of temporal vector sequences. We developed an advanced version of RISq and applied it to speech recognition, a task most commonly performed with Hidden Markov Models (HMMs) or Dynamic Time Warping (DTW). RISq is substantially different from both these methods and presents several advantages over them: robust recognition can be achieved using only a few samples from the input sequence and training can be carried out with one or more examples per class. This enables much faster training and also allows to recognize speech with a variety of accents. A two-step classification algorithm is used: first the training samples closest to each input sample are identified and weighted with a parallel algorithm (indexing). Then a maximum weighted bipartite graph matching is found between the input sequence and a training sequence, respecting an additional temporal constraint (sequencing). We discuss the application of RISq to speech recognition and compare its architecture and performance with that of Sphinx, a state-of-the-art speech recognizer based on HMMs.**

Keywords: example-based speech recognition; recognition by indexing and sequencing; RISq; compounded examples

I. Introduction

Hidden Markov Models (HMMs) are one of the most popular techniques for classification of temporal sequences, such as speech [1], gestures [2] and human actions [3]. HMMs are a parametric technique which uses statistical models to represent the underlying process. They need to maintain a very large number of parameters, which in turn leads to a long and complicated Expectation-Maximization (EM) training algorithm that requires a large amount of data. In most HMM-based state-of-the-art speech recognizers, such as Sphinx [4], the acoustic model for a word is composed of smaller models of the phonemes in the word. Phonemes in turn are composed of three states, an on-set, a middle, and an end, and each of these states is modeled with a Gaussian mixture model (GMM). When D dimensions (i.e. features) and G Gaussians are used, $O(D^2 \times G)$ parameters are needed for each state if full covariance matrices are used. This can be reduced to $O(D \times G)$ when only elements on the main diagonal of the covariance matrices are retained. In Sphinx, $D = 39$ and G varies between 8 and 32, which represent typical values for this kind of system. A complex and data-intensive training procedure is needed to reliably estimate these parameters.
Dynamic Time Warping (DTW) is a simpler non-parametric technique to align sequences that was also applied to speech recognition. In its original formulation, DTW can only be applied to two mono-dimensional vectors: first a matrix is built to measure the distance from each element of one sequence to each element of the other one; then a minimum-cost path is found with a dynamic programming algorithm. Only recently DTW has been extended to multi-dimensional vectors [5] and multiple sequences [6].
Recognition by Indexing and Sequencing (RISq) [7, 8, 9] is also a non-parametric technique that takes a classical pattern recognition example-based approach modified for vector sequencing and presents some advantages with respect to both HMMs and DTW. RISq uses a simple training procedure and can achieve robust recognition even when training is performed with only one example sequence from each class. However, it is possible to train RISq with many independent example sequences for the same class. In this case, classification uses a compounded example approach, where parts of different example sequences can be combined to optimally match the input sequence. These advantages are particularly critical for applications where only a few sequences are available for training and where significant differences are present among data of the same class, such as in a multi-modal effective communication interface for the elderly [10] that we are developing.
While RISq has some resemblance to DTW, it also has significant improvements over it. Both methods match sequences while allowing warping and use dynamic programming to find the optimal matching score. However RISq and DTW are different in many respects:
1) Distances: DTW minimizes the distance between sequences, whereas RISq maximizes the total similarity, which leads to different outcomes. Moreover, RISq can also penalize dissimilarity, while DTW can only use distance penalties.
2) Segmentation: With DTW the beginning and end of the sequence must be defined. When applying DTW to continuous data streams, such as speech, this requires segmentation, a very error-prone process. RISq does not need boundary-aligned sequences as it can match only parts of a sequence.

3) Matching-1: With DTW a continuous monotonic path must be built to match each sample in one sequence to a sample in the other sequence. This also means that DTW has difficulties handling incomplete sequences due to signal noise and interference. By performing bipartite graph matching, RISq allows partial matching of sequences in any sequential configuration.
4) Matching-2: indexing in RISq allows partially parallel recognition, whereas DTW is entirely serial.
In principle, RISq can recognize any kind of spatial or temporal sequence, such as human activities, to which it was already successfully applied [7]. The original formulation of RISq is described in [8], which also suggests its possible application to speech. However, it neither clearly describes the methodology nor gives all the necessary details for a working application for speech. The major contributions of this paper are improvements to the basic methodology and the application of RISq to both isolated-word and continuous speech recognition. The performance of RISq is compared to that of Sphinx on a standard speech database.

II. Related work

The only application of RISq [8] so far was in the activity recognition domain [7, 11]. In this work 8 different activities were used for training and testing reaching an accuracy of 100% in ideal conditions. In order to improve robustness to occlusion, the human body was decomposed in 5 parts: head and torso, arms and legs. The position and speed of head and torso, upper and lower arms and upper and lower legs were tracked, using a total of 18 dimensions. Votes were collected independently for each body part, so that recognition rates of 70% to 97.5% were obtained even with up to 3 out of 5 occluded body parts.
Sphinx is a speech recognition system based on HMMs, developed by Carnegie Mellon University over the last two decades, and is currently considered the state-of-the-art; an overview of the most recent version, Sphinx-4 is given in [4].
Recently, there has been renovated interest in example-based systems for speech recognition. One of the main reasons that initially lead to the development of HMMs was the very limited available computational power. As computers grow more powerful, example-based techniques become feasible even for medium and large vocabulary applications.
The example-based system presented in [12] drops modeling but keeps the Bayesian approach that is typical of HMMs. The main characteristics of this system are a new class-based distance measure, a bottom-up template selection algorithm, the use of costs based on meta-information and an improved DTW decoder. While this approach does not outperform HMMs alone, it improves on state-of-the-art results when combined with HMMs.
Another interesting method is based on sparse representation [13], i.e. representing a signal by a linear combination of example units. The authors investigate several alternative ways of incorporating sparse representation in classical HMM systems. They show how one of the proposed techniques, dubbed sparse classification, outperforms HMMs especially in conditions of very noisy speech by separately accounting for speech and noise.

III. Methodology

RISq is a non-parametric technique, so it does not make any attempt at building a model for each class to be recognized. Instead, training is performed by simply storing one or more example sequences per class in an underlying data structure. Each sample in a sequence is a vector, whose kind and dimensionality varies with the application. For some applications it might be possible to directly use raw data, whereas with other ones, such as speech, preprocessing is necessary to extract features.
After training is performed, an unknown input sequence can be classified using a two-step algorithm. The first step is indexing, which consists in identifying the training samples closest to each input sample and assigning them a vote dependent on their distance. The second step is sequencing, which finds a maximum weighted bipartite graph matching between the input sequence and a training sequence, respecting an additional temporal constraint.

A. Training procedure

With RISq, similarly to DTW, the training procedure simply consists of storing all the samples from the training sequences in an underlying data structure, along with its timing, class and a sequence identifier. This training procedure can be performed with one or more multiple independent sequences per class, an approach that cannot be used with HMMs or DTW.
With our approach, in the classification stage we retrieve the training samples that are closest to some of the samples from the input sequence, according to some distance function. This task is known in computational geometry as a range search and it is related to nearest-neighbor search. When operating in more than a few dimensions, the feature space is sparse and the most efficient known data structure for this task is a kd-tree (k-dimensional tree). If n points are stored in a balanced kd-tree, the computational complexity of a range search is $O(n^{1-1/d} + p)$, where p is the number of points returned [14]. This result holds if $n \gg 2^D$, where D is the number of dimensions, otherwise most of the tree needs to be searched and the complexity is no better than exhaustive search. Despite suffering from the so-called “curse of dimensionality”, the kd-tree still provides a modest improvement with respect to exhaustive search and is the best known exact method for nearest neighbor retrieval in a high-dimensional space.

B. Classification procedure

The classification procedure comprises three main steps: downsampling the input sequence, indexing and optimal sequencing.
Supposing that the input sequence is initially sampled at the same frequency as the training sequences, we can resample it at a much slower rate, reducing the computational burden of the subsequent stages without significantly compromising the recognition rate [7]. The re-sampling can be performed with uniform sampling or with random sampling from a uniform distribution. The second case is more interesting, as it simulates the situation of an input sequence with missing data.

In the next step each input sample is indexed in the kd-tree: a nearest-neighbor search is performed to retrieve a fixed number of its nearest training samples and each of them is assigned a vote which is an inverse function of the Euclidean distance from the corresponding input sample. This constitutes a difference with our previous approach of performing a range search, where a range is specified and a variable number of neighbors is returned at each sample. The search with a fixed number of neighbors provides a better recognition rate and less variability in the time needed for recognition. This stage is parallel since it considers all of the samples from each example sequence and class at the same time, ignoring the class labels and timing attached to each sample that will be taken into account during the sequencing stage. We assign a vote to each retrieved neighbor proportionally to a Gaussian function of the distance d: $vote = e^{-d^2/2\times\sigma^2}$ where σ is a parameter selected with statistical analysis. This leads to votes between 0 and 1.

The last step in the classification is to find an optimal sequence of votes for each class, that is a maximum weighted bipartite graph matching between the samples of the input sequence and the retrieved samples of each training sequence, a problem efficiently solvable by the Hungarian method [15]. However, we apply an additional temporal sequencing constraint that imposes the same ordering on the matched training sequence as that of the input sequence. Thus, the standard weighted bipartite graph matching no longer applies and we need to develop a novel algorithm to efficiently solve this problem. Let us show an example.

We will first consider the case where training is performed with only one example sequence for each class. Assume that we are computing an optimal sequence for class i and let us define an input sample as t_p and a training sample as $n_{i,q}$, where p and q represent timing. Then the constraint is: if t_a votes for $n_{i,c}$ and t_b votes for $n_{i,d}$ and $a < b$, then $c \leq d$. An example of this procedure is shown in Figure 1.

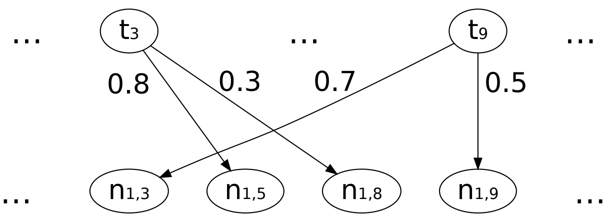


Figure. 1: Sequencing example

Suppose that we are computing the matching score between the input sequence and class 1. At the top of the figure we see two samples from the input sequence, t_3 and t_9. At the bottom of the figure we see training samples voted by the two input samples from the example sequence for class 1. The numbers on the edges represent the corresponding votes. The maximum matching is $\{t_3, n_{1,5}\}, \{t_9, n_{1,3}\}$, yielding a score of 1.5. However, this is not allowed because t_3 precedes t_9, but $n_{1,5}$ follows $n_{1,3}$. Therefore, the optimal matching that respects the temporal constraint is $\{t_3, n_{1,5}\}, \{t_9, n_{1,9}\}$, yielding a score of 1.3.

When classifying an input sequence, we need to compute an optimal sequence for each trained class, with the following algorithm: at each training sample for that class voted by each input sample, we save the optimal sequence ending at that training sample. In order to efficiently find optimal sequences, we can apply a dynamic programming approach, since this problem exhibits optimal substructure. The optimal sequence ending at some training sample $n_{i,d}$ is formed by the vote towards that training sample and the optimal sequence ending at some previous training sample $n_{i,c}$ that does not violate the temporal constraint (if one exists). Therefore at each training sample, we can save the optimal sequence ending with that sample and look it up later (memoization). This greatly simplifies the process of obtaining optimal sequences. Pseudocode of the classification procedure is in Figure 2.

```
1: {$C$: number of trained classes}
2: {$S$: number of samples selected from input sequence}
3: {$N_i$: number of different training times among all samples voted for with class $i$}
4: Initialize $neighbors$ data structure, with size $C \times S$
5: for $j = 1$ to $S$ do
6:   $sampleNeighbors$ = kd-treeQuery($t_j, numNeighbors$)
7:   for $k = 1$ to length($sampleNeighbors$) do
8:     Compute vote from $t_j$ to $sampleNeighbors(k)$
9:     Add $sampleNeighbors(k)$ to $neighbors(sampleNeighbors(k).class, j)$
10:  end for
11: end for
12: for $i = 1$ to $C$ do
13:   Initialize $classSeq$ data structure, with length $N_i$
14:   for $j = 1$ to $S$ do
15:     $myNeighbors = neighbors(i, j)$
16:     Initialize $sampleSeq$ data structure, with length $k$
17:     for $k = 1$ to length($myNeighbors$) do
18:       {Following loop finds maximum sequence ending at a training time less than the training time of this retrieved sample}
19:       $maxSeq.vote = 0$
20:       for $w = 1$ to $N_i$ do
21:         if $classSeq(w).lastTrainTime < myNeighbors(k).trainTime$ then
22:           if $classSeq(w).vote > maxSeq.vote$ then
23:             $maxSeq.vote = classSeq(w).vote$
24:           end if
25:         else
26:           Exit loop
27:         end if
28:       end for
29:       $sampleSeq(k).vote = maxSeq.vote + myNeighbors(k).vote$;
30:     end for
31:     for $k = 1$ to $length(myNeighbors)$ do
32:       Add $sampleSeq(k)$ to $classSeq$
33:     end for
34:   end for
35:   Save optimal sequence for this model
36: end for
```

Figure. 2: Pseudocode of indexing and sequencing

Once this process is completed, the final vote for each class is given by the sum of votes in the optimal sequence and the input sequence is classified according to the class with the highest vote.

The analysis of the algorithm in Figure 2 reveals that the greatest contribution to time complexity comes from the sequencing stage. The execution time is directly proportional to the number of trained classes C, the number of samples S selected from the input sequence, the number of retrieved neighbors per sample T and the number of different training times among all samples voted for with class i. Therefore the worst case temporal complexity of the algorithm is $O(C \times S \times T \times \max_i N_i)$. In order to further reduce the average case time complexity when we will extend RISq to a much larger number of classes, we will need to limit the sequencing stage to the most likely classes. This could be done based on the information collected in the voting stage, such as number and magnitude of votes collected per class, and domain knowledge, such as a language model for speech recognition.

When training with multiple examples per class, a compounded example approach is used. Sequencing works as just described, however it is performed separately for each example for that class. Subsequently, an additional step is performed to merge matchings from different examples into an optimal sequence. This procedure must take care of preserving the timing information since different example sequences for the same class might have different lengths. Therefore the timing of the input sequence is used to guarantee synchronization. If the same sample from the input sequence voted for two or more samples from different training examples, then the one with the highest vote is selected to belong to the final sequence for that class.

C. *RISq for continuous data streams*

So far we have described the RISq methodology as applied to isolated events, such as single words or gestures. However, it is useful for many domains to be able to apply RISq to a continuous data stream, such as video, audio or some other kind of signal. This adds complexity, since we also need to parse out individual events from the data stream before we can classify them. The main idea to adapt RISq to continuous data streams, such as continuous speech, consists of three major steps: segment the data stream; score each segment using the methodology for isolated sequences; post-process the votes from each segment to decide when to emit class labels.

The first step consists in segmenting the data stream. Ideally, each segment would correspond to a single event and the classification on the data stream would not be more complicated than in the isolated case. However, this does not happen in practice as the segmentation task itself is often extremely difficult. In the case of speech, for example, this is due to the fact that many parts of the speech have very low energy, so that an energy-based approach is not optimal, and also to the fact that words are often uttered very close to each other, often with no pause between them. In fact, one task at which humans excel is being able to parse out single words from the stream, a significant challenge when learning a new language.

However, RISq is not particularly susceptible to the segmentation algorithm, because it can easily match parts of an event, without needing the whole event for classification. This allows the use of a simple segmentation algorithm, without a large degradation in performance. We are currently employing a technique where the stream is segmented in fixed-length overlapped segments.

After segmentation, each segment is scored against the training classes using the methodology for isolated sequences as described in the previous section.

Finally, it is necessary to post-process the votes from each segment to decide when to emit class labels. This is necessary because many subsequent segments can be part of the same event, so we cannot just insert in the recognized stream many occurrences of the same event, as this would lead to a so-called insertion error. On the other hand, if two (or more) events are covered by only one segment, we will miss one of them, causing a deletion error. Finally, a substitution error happens when an event is misclassified.

This post-processing phase is currently performed with a simple peak detection technique. This is motivated by the fact that as a segment "slides" through the event, the overlap between the segment and the event will increase up to a maximum and then decrease. Therefore, if we plot the votes from subsequent segments for the corresponding class, the chart will exhibit a peak. When this peak is reached, we will then emit a corresponding class label. Some work remains to be done in this area as well, as we might try to find a more efficient post-processing technique. An example of the segmentation and post-processing for speech is in Section V.

IV. Results on isolated speech

In order to evaluate results with RISq and Sphinx we performed tests with part of the data from the Center for Spoken Language Understanding (CSLU) Speaker Recognition corpus [16], which includes 8KHz telephone speech recorded from 91 speakers. Each speaker called an automated system 12 times over 2 years, answering free-speech questions and pronouncing predefined words.

From this corpus, we chose 2 predefined 5-digit sequences that each speaker pronounced 4 times during each call: "five three eight two four" and "six one oh nine seven". Thus our dictionary consists of these 10 different digits, the same as in a digit recognition demo provided with the Sphinx system.

Each sequence was manually segmented to isolate the single digits. Given that we used only the data from the first call for each speaker, our dataset consisted of 91 speakers, each pronouncing 10 digits for 4 times, for a total of about 3,500 digits, since a few data are missing from the database.

In order to assess the performance of both methods, input sequences were classified and results were collected in a multiclass confusion matrix, where rows corresponded to actual classes and columns to estimated classes. However, computing statistics from the confusion matrix and performing Receiver Operating Characteristic (ROC) analysis with multiple classes is significantly harder than in the binary case, generally leading to a complexity exponential in the number of classes for an exact solution. Only recently a good approximation has been proposed in the form of a pairwise analysis [17].

Since this kind of analysis is not significant for our application, we follow a more direct approach by computing the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) on a per-class basis [18]. Suppose that we have 3 classes and we want to compute the statistics for class 1; we would then regard the values in the confusion matrix as follows:

$$\begin{bmatrix} TP & FP & FP \\ FN & TN & TN \\ FN & TN & TN \end{bmatrix}$$

Similar considerations hold for class 2 and 3. On the basis of these values, we compute four statistics, again on a per-class basis: hit rate (TP / TP+FN), false alarm rate (FP / FP+TN), positive predictive value (TP / TP+FP) and negative predictive value (TN / TN+FN). This allows to plot one point in ROC space for each class. In order to compute full ROC curves on a per-class basis, we first collect the final votes towards each class for each input sequence and then regard all of the sequences whose real class is the class we are considering as positive examples and all of the sequences whose real class is different as negative examples, as shown above. Full ROC curves for each class are computed using a standard algorithm [19] and averaged to yield one ROC curve for the classifier.
The Real Time Factor (RTF), defined as the ratio of the time needed to process an input sequence to the duration of the sequence, was measured on a MacBook Pro with a 2.4 GHz Intel Core 2 Duo CPU and 4 GB RAM. We used the Sphinx 4 standard Java version and our Matlab implementation for RISq. The kd-tree implementation is in C. We made no major attempt at optimizing the RTF yet.

A. Preprocessing of speech data

In speech recognition, preprocessing of data is necessary to extract significant features from the audio stream. In order to have a fair comparison with Sphinx, we used the preprocessor included in Sphinx to extract Mel-Frequency Cepstral Coefficients (MFCCs), the most widely used kind of feature. The MFCCs extracted from each sequence were used as input data for both RISq and Sphinx. In the preprocessing, the original utterance is processed with sliding overlapped Hamming windows and 13 MFCCs coefficients are extracted from each window, along with their first- and second-order derivatives, leading to a total of 39 features. This constitute a vector that corresponds to a sample in our sequence, i.e. a 39-dimensional vector. The typical duration of a word in our tests varies from 0.5 seconds to 1 second, which translates to about 50 to 100 samples per word when using the recommended parameter of 25.625 ms for a window and a gap of 10 ms between windows. The full procedure is shown in Figure 3.

B. Results with RISq

RISq is using a subset of 24 features of the 39 extracted by the Sphinx pre-processing engine. We excluded the first feature (a DC component) and the second derivatives, since we did not observe any significant improvement on the recognition rate when also using them. Training was performed as described in Section II.A. RISq is very flexible with respect to the size of the example sequences to use. In all the experiments mentioned in this paper, we trained the algorithm using full words, also given the limited size of the dictionary. Alternatively sub-word units such as syllables or phones could be used. Classification was performed as explained in Section II.B, using compounded examples.
In Table 1 we show sample results obtained with both uniform and random sampling depending on the percentage of samples used from the original sequence. These results are obtained from training with 1 example digit per speaker for 10 speaker, and testing on 10 different speakers.

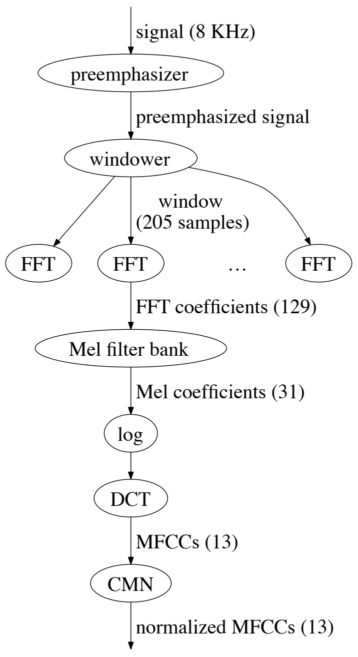


Figure. 3: Extraction of MFCCs from speech data. The acronyms are respectively: Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT) and Cepstral Mean Normalization (CMN).

Samples	HR (unif.)	HR (rand.)	Seq. time (ms)
10%	77.6%	76.7%	113
25%	81.7%	79.8%	163
50%	90.9%	89.9%	244
75%	91.4%	90.5%	326
100%	92.0%	91.2%	408

Table 1: Results when varying sampling rate

It can be seen that the recognition rate improves significantly only until 50% of the samples are used, while the time increases about linearly. Based on this result, we use 50% of the samples in all the subsequent experiments. Also, the hit rate for the random sampling is on average only 1.1% lower than with the uniform sampling. This is an important result, as the random sampling simulates the situation of missing data in the input sequence.
The other main parameters involved in the classification are the number of returned nearest neighbors k and the voting σ.

Optimal values of these parameters, determined by statistical analysis, allow to greatly reduce the time needed for classification, while maintaining a high recognition rate. In particular we used $k = 20$ for each sample and $\sigma = 0.5$.
We setup three different experiments. In the first experiment, we trained RISq with some data from one speaker and tested with different data from the same speaker. In this experiment we tested the effect of increasing the number of examples in the training. When using one, two or three examples per speaker we obtained 97.7%, 98.1% and 98.5%. We consider a very noticeable achievement to reach such a high recognition rate even with only one example per speaker.
This procedure was repeated for each speaker and the average ROC curve obtained with the process described earlier is shown in Figure 4. To improve the visibility of the differences in the average ROC curves, which are difficult to discern in the full ROC curve in Figure 4, the top left quadrant of the same chart is magnified in Figure 5. Also, Table 2 shows the per-class and average true positive rates when the false positive rate is 0.1. The RTF using the values of the parameters indicated earlier was 0.14.

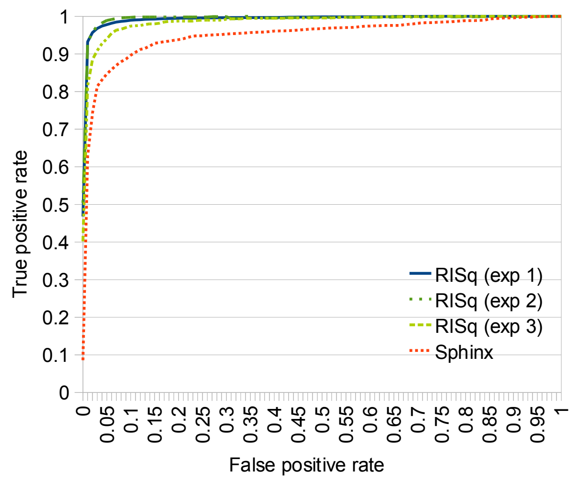


Figure. 4: Average ROCs

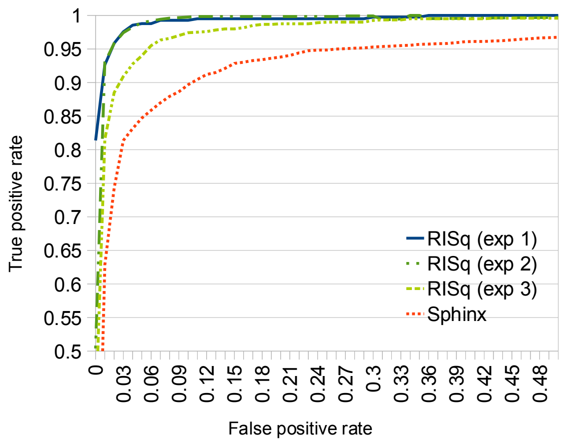


Figure. 5: A magnified version of the top left quadrant of Figure 4

	RISq1	RISq2	RISq3	Sphinx
0	.95	.99	.99	.89
1	.99	.99	.99	.73
2	.99	.99	.99	.98
3	.99	.99	.99	.96
4	.99	.99	.98	.95
5	.99	.99	.91	.87
6	.99	.99	.90	.67
7	.99	.98	.98	.99
8	.99	.99	.99	.93
9	.99	.99	.99	.99
avg	.99	.99	.97	.89

Table 2: Per-class TPR with FPR=0.1

In the second experiment, we trained RISq with multiple examples for each word from multiple speakers and tested with different sequences from the same speakers. Results given in Figure 4 and Table 2 are very similar to those obtained in experiment 1. In particular, the recognition rate for the digit "oh" improved from 95% to 99% thanks to the multiple training sequences. The RTF was 0.22.
In the third experiment, we trained RISq with data from a varying number of speakers and tested with data from 10 different speakers. For a fair comparison, the testing set remains the same. We obtained respectively 90.9% recognition rate with 10 speakers in the training set, 91.9% with 20 speakers, 92.4% with 30 speakers and 93.9% with 40 speakers. Again, it is remarkable that we obtained 90.9% when training using only one example per word from 10 speakers.
The ROC curve in Figure 4 is for 40 speakers in the training set and it shows that RISq outperforms Sphinx even in this more challenging task. The RTF was 0.30.
Note that the flexibility of the training procedure easily allows both speaker-dependent and speaker-independent speech recognition. Experiment 1 is an example of speaker-dependent recognition, whereas experiments 2 and 3 perform speaker-independent recognition and they show excellent results with this small dictionary.

C. *Results with Sphinx*

The training procedure with Sphinx is more elaborated than it is with RISq and requires a much larger amount of training data. Therefore, we used the standard acoustic models trained with 8KHz speech from the Wall Street Journal (WSJ) provided with Sphinx and we adapted them to the same 40 speakers used for training in RISq.. The dictionary consisted of the 10 digits and the language model was a simple grammar with each sentence formed by only 1 of the digits. The system was configured with a flat linguist and a simple breadth first search manager, the suggested architecture for this kind of task. Even if the WSJ database contains a large number of acoustic models, only those for the 10 words in the dictionary are considered during classification.
Testing was performed on the same 10-speaker dataset as for RISq. The average ROC curve shown in Figure 4 proves slightly worse than that obtained with the third experiment performed with RISq. Results in Table 2 show significant differences in the recognition rate among classes. In particular, recognition of "one" and "six" was significantly worse with Sphinx, possibly due to silence assumptions in the WSJ acoustic models. The RTF was 0.02.

V. Results on continuous speech

Tests on continuous speech were also performed using the CSLU database. Evaluating the recognition rate of a continuous-word speech recognizer is not straightforward, because the number of words in the utterance and in the recognized string can differ. Therefore it is necessary to first align the reference text and the automatic transcription. Subsequently, it is possible to compute the Word Error Rate $WER = \frac{S+D+I}{N}$ and Word Accuracy $WAcc = \frac{N-(S+D)}{N}$. S is the number of substitutions, i.e. words which has been misclassified. D is the number of deletions, i.e. words present in the reference text but not in the automatic transcription. I is the number of insertions, i.e. words present in the automatic transcription but not in the reference text.

A. *Results with RISq*

Training and classification were performed as explained in Section II. Figure 6 shows example results of applying RISq to continuous speech.

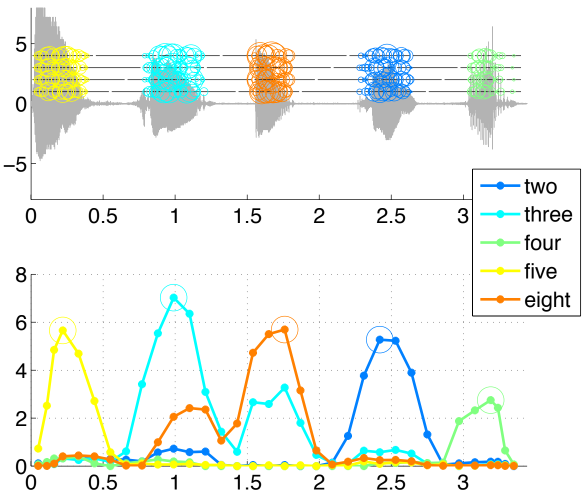


Figure. 6: Example of RISq applied to continuous speech on the input sequence “five three eight two four” uttered by speaker 38 in the CSLU database. In the upper part, the speech waveform is in gray in the background. The overlapped segments are in black. The matching scores are depicted as circles with different colors and diameters. The colors represent the words (classes). The diameters are proportional to the matching scores of the input samples used in RISq. In the lower part we show the total matching scores for the words where colors identify words as above. The circles here are automatically identified by our peak detection algorithm and represent recognized words.

In the example results from Figure 6, the training data consisted of a different utterance of the same five words by the same speaker. The figure shows overlapped segments with duration of 0.4 seconds and 0.1 second gap. The circles on each segment represent the testing samples from the input sequence matched to the winning training class. Colors identify the class for each segment. Therefore we can see that each segment has been correctly classified.

The bottom part of Figure 6 shows the votes corresponding to each segment. For easier reference, the votes are plotted aligned with the middle of the segment in the top part of the figure. It is possible to clearly see the peaking behavior of votes as described in the previous section. In order to detect peaks we adopt the following algorithm. At each time step, we consider the maximum vote. If the vote is greater than all the other votes at the previous and following time steps, then a peak is detected and the corresponding class label is emitted. In this example, the automatic transcription is “five three eight two four” so we do not have any insertion, deletion or substitution error.

We setup the same experiments as for isolated speech. In the first experiment, we trained RISq with some data from one speaker and tested with different data from the same speaker, obtaining a 96% word accuracy. In the second experiment, we trained RISq with multiple examples for each word from multiple speakers and tested with different sentences from the same speakers, obtaining a 94% word accuracy. In the third experiment, we trained RISq with data from several speakers and tested with data from different speakers. This yielded a 91% recognition rate with a RTF of 0.5.

B. *Results with Sphinx*

For tests on the CSLU database, the dictionary consisted of the 10 digits and the language model was a simple grammar with each sentence formed by only 1 of the digits. Even if the WSJ database contains a large number of acoustic models, only the models for the words in the dictionary are considered during classification. The system was configured with a flat linguist and a simple breadth first search manager, the suggested architecture for this kind of task. We obtained a word accuracy of 91%.

VI. Conclusions and future work

We have described an improved methodology for RISq and its application to both isolated-word and continuous speech recognition. By following a sequenced pattern recognition approach, RISq eliminates the need to maintain a very large number of parameters and a complicated training procedure to estimate them. We have compared RISq to Sphinx, a state-of-the-art speech recognizer based on HMMs. The results that we obtained with RISq proved promising and better than those obtained with Sphinx, despite the fact that RISq is a much simpler and younger method. However, the comparison holds so far only for such a small task as has been discussed in this paper. Sphinx is currently a large vocabulary continuous speech recognizer and RISq cannot yet handle the same complexity.

We showed that RISq is able to perform well on independent-speaker speech recognition just by training with a limited number of multiple independent example sequences from different speakers, instead of building acoustic models. We are currently working on improving recognition especially for continuous speech, as well as on increasing the size of our dictionary. Our method has the potential for a significant impact on state-of-the-art speech recognition, especially in those domains where fast adaptation to new users is required, such as assistive technology for the elderly.

References

[1] L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” *Proc. of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[2] J. Yang and Y. Xu, “Hidden Markov Model for Gesture Recognition,” Carnegie Mellon University, Robotics Institute, Technical report CMU-RI-TR-94-10, 1994.

[3] J. Yamato, J. Ohya, and K. Ishii, “Recognizing Human Action in Time-Sequential Images Using Hidden Markov Model,” in *Computer Vision and Pattern Recognition*, 1992, pp. 379–385.

[4] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel, “Sphinx-4: A Flexible Open Source Framework for Speech Recognition,” Sun Microsystems, Technical report TR-2004-139, 2004.

[5] G. A. ten Holt, M. J. T. Reinders, and E. A. Hendriks, “Multi-Dimensional Dynamic Time Warping for Gesture Recognition,” in *Annual Conference of the Advanced School for Computing and Imaging*, 2007.

[6] N. U. Nair and T. V. Sreenivas, “Multi Pattern Dynamic Time Warping for Automatic Speech Recognition,” in *Tencon*, 2008.

[7] J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram, “Human Activity Recognition Using Multidimensional Indexing,” *PAMI*, vol. 24, no. 8, pp. 1091–1104, 2002.

[8] J. Ben-Arie, “Method of Recognition of Human Motion, Vector Sequences and Speech,” US Patent 7,366,645, April 2008.

[9] S. Franzini and J. Ben-Arie, “Speech Recognition by Indexing and Sequencing,” in *Proceedings of the International Conference of Soft Computing and Pattern Recognition*, 2010, pp. 93–98.

[10] M. Zefran, J. Ben-Arie, B. Di Eugenio, and M. D. Foreman, “Effective Communication with Robotic Assistants for the Elderly: Integrating Speech, Vision and Haptics,” NSF Grant #0905593, 2009.

[11] D. M. Sivalingam, “Analysis of Human Motion: Labeling, Activity and Multi-class Recognition,” Master’s thesis, University of Illinois at Chicago, Department of Electrical and Computer Engineering, 2003.

[12] M. DeWachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. VanCompernolle, “Template-Based Continuous Speech Recognition,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, pp. 1377–1390, 2007.

[13] J. F. Gemmeke, T. Virtanen, and A. Hurmalainen, “Exemplar-Based Sparse Representations for Noise Robust Automatic Speech Recognition,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, pp. 2067–2080, 2011.

[14] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[15] H. W. Kuhn, “The Hungarian Method for the Assignment Problem,” *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

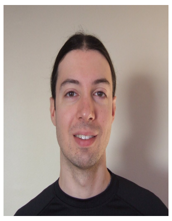
[16] R. Cole, M. Noel, and V. Noel., “The CSLU Speaker Recognition Corpus,” in *International Conference on Spoken Language Processing*, November 1998.

[17] T. C. W. Landgrebe and R. P. W. Duin, “Approximating the Multiclass ROC by Pairwise Analysis,” *Pattern Recognition Letters*, vol. 28, pp. 1747–1758, 2007.

[18] G. S. Rees, W. Wright, and P. Greenway, “ROC Method for the Evaluation of Multi-class Segmentation/Classification Algorithms with Infrared Imagery,” in *BMVC*, 2002, pp. 537–546.

[19] T. Fawcett, “An Introduction to ROC Analysis,” *Pattern Recognition Letters*, vol. 27, pp. 861–874, 2006.

Author Biographies



Simone Franzini received his BS and MS in Computer Science Engineering from Politecnico di Milano, Italy, in 2004 and 2006. He also obtained a MS in Computer Science in 2007 from University of Illinois at Chicago, where he is currently a PhD candidate. His research interests are in Pattern Recognition, Image Analysis, Computer Vision, Mobile Robotics and Artificial Intelligence.

Jezekiel Ben-Arie received the B.Sc., M.Sc., in Electrical Engineering and in 1986 PhD. Degree in Aerospace Engineering from the Technion, Israel Institute of Technology, Haifa. In 1995 he joined the ECE Dept. of UIC and established there the Machine Vision Lab of which he is the director. Currently, he is a Professor in the Electrical and Computer Engineering and Computer Science Departments, University of Illinois, Chicago.
Professor Ben-Arie contributed significant works in the areas of Computer Vision, Signal and Image Processing, Image and Video Analysis, Object, Target and Speech Recognition, Human Hearing, Human Motion Analysis, Neural Networks, Biometrics and Information Theoretic Text Summarization. His research resulted in more than 140 scientific publications. So far, Prof. Ben-Arie has successfully completed 19 funded research projects by NSF, DARPA, ONR, Whitaker Foundation and others.