Swedish Speech Recognition Using Linear Time Normalization and Feature Selection Optimization

Mattias Wahde
Chalmers University of Technology
Department of Applied Mechanics
Göteborg, Sweden
Email: mattias.wahde@chalmers.se

***Abstract*—A system for Swedish isolated word recognition has been developed, intended for use in an intelligent news reader agent for elderly care. The system uses linear time normalization of feature time series, as well as optimization (with a genetic algorithm) of both feature selection and feature weighting. The optimization of feature selection results in a potentially important decrease in the time needed to recognize a spoken word, while maintaining speech recognition performance.**

Keywords-Speech recognition; optimization; linear time normalization

I. INTRODUCTION

The increase in the fraction of elderly people, relative to the working population, is a strong demographical trend at present: One recent estimate expects the fraction of elderly people (65+ years old) in the European Union to increase from around 17% in 2010 to 30% in 2060 [1]. Furthermore, the fraction of people above the age of 80 is expected to increase from around 5% to 12%, over the same time span.

In the near future, it is therefore likely that a variety of technological tools, such as assistive or partner robots [2], as well as intelligent homes [3], will come to play an important role in elderly care. Indeed, some prototypes already exist, such as the therapeutic seal robot Paro [4] and the assistive robot Kompai [5]. An increased role of such tools is probable for several reasons. For example, the (relative) decrease in the size of the working population, combined with the increase in the size of the elderly population, is likely to cause staff shortages in elderly care. Furthermore, as a quality-of-life issue, many elderly people prefer to live in their own home (rather than in, say, a nursing home) for as long as possible [2], something that can be facilitated using technological tools that, for example, can monitor medicine intakes, alert relatives or healthcare workers in case of injuries etc.

In order for such technological tools to be applicable in meaningful interactions with elderly people, they will need to be equipped with intuitive user interfaces, as one cannot expect their users to be familiar with computers, let alone robots. In addition, robots and agents must, of course, be able to interact in their users' language, which might not be one of the major languages, such as English, Japanese, or Chinese. For these reasons, the Adaptive systems research group at Chalmers University of Technology, in Göteborg, Sweden, has recently started a project (that will be described in detail elsewhere) to develop both intelligent software agents and a partner robot intended for use in elderly care in Sweden.

This paper will consider one particular aspect of that project, namely speech recognition (in Swedish) for an intelligent news reader agent that can access online news based on the user's commands. Speech recognition (henceforth: SR) is less developed (than in English) in languages spoken either by rather few people [6] or in emerging countries; see, for example, [7]. In fact, much of the SR-related robotics research is focused on English, since it is spoken by almost all researchers, regardless of nationality, but perhaps not by all people in the general population, and certainly not all elderly people. Hence, improvement of SR systems is an important step towards the development of technological tools for elderly care in countries where English (or any other major language) is not the first language.

Of course, the problems and issues encountered when developing an SR system are similar, regardless of the language considered. In general, two main forms of SR can be distinguished, namely (i) isolated word recognition (IWR), and (ii) continuous speech recognition (CSR) [8]. Over the years, SR has been approached using many different techniques, in particular dynamic time warping (DTW) (see, e.g., [8], Chapter 4) which involves a classification method based on comparison of time series of different length, and hidden Markov models (HMMs) (see, e.g., [8], Chapter 6) that, effectively, constitute a stochastic (probabilistic) extension of DTW. The HMM approach currently dominates SR research, and most of the state-of-the-art SR systems employ this method. However, deterministic approaches (such as DTW) are also useful, particularly in command-style SR systems focusing on IWR with a rather limited vocabulary [9]. DTW attempts to find the optimal alignment between time series (containing, for example, features extracted from sound data) while, simultaneously, computing a distance measure between the two series. Thus, comparing the feature time series extracted from a given utterance to feature time series stored for template sounds, one can, based on the minimum distance found, determine which word was spoken. Even though DTW is frequently applied in deterministic SR systems, some recent work, further discussed in Section IV, has indicated that DTW does not, in fact, necessarily improve performance when matching time series of different length.

 ISBN: 978-1-61208-224-0

Thus, in this paper, a simpler approach (for Swedish SR) will be evaluated, in which sound features are first computed from partially time-overlapping frames extracted from a spoken word. The time series thus obtained are normalized to unit length, and are then resampled at equidistant (relative) times, making direct comparisons between different time series (obtained from different utterances) possible, without DTW. Several time series are generated for each sound, corresponding to different sound features, and the weighted Euclidean distance between the features from the reference sounds (recorded *a priori*) and the currently spoken word is then used as a measure of dissimilarity. Furthermore, the *selection* and weighting of the features used for SR have been optimized using a genetic algorithm (henceforth: GA).

The structure of the paper is as follows: The method is described in Section II. The results are given in Section III and are discussed in Section IV. The conclusions are presented in Section V.

II. METHOD

The SR method used here, which has been implemented in C# .NET, consists of four main parts: First, sounds are preprocessed and divided into short sound frames, using a fairly standard procedure. Next, features are extracted from the sound frames. Then, in SR, the features obtained are compared to features extracted from template words, in order to determine which word was spoken. The final part involves optimization of feature selection and feature weighting.

A. Sound preprocessing

The first preprocessing step is to subtract the mean from the sound samples (which, for 16-bit sounds, range from -32768 to 32767), i.e., removing any static (DC) components from the sound. The next step is to extract the word, assuming that a single word was spoken. This is done by first moving forward along the sound samples, starting from the μ^{th} sample, and forming a moving average involving (the modulus of) μ sound samples. Once this moving average exceeds a threshold t_p, the corresponding sample, with index k_s, is taken as the start of the word. The procedure is then repeated, starting with sample $\nu-\mu+1$, where ν is the number of recorded samples, forming the moving average as just described, and then moving backward, towards lower indices. When a sample (with index k_e) is found for which the moving average exceeds t_p, the end point has been found. The sound containing the $k_\text{e}-k_\text{s}+1 \equiv m$ samples is then extracted and is henceforth referred to as a *word*. The word is then pre-emphasized. In the time domain, the pre-emphasis filter takes the form

$$s_k \leftarrow s_k - cs_{k-1}, \tag{1}$$

where s_k denotes the k^{th} sample and c is a parameter with a typical value slightly below 1. As is evident from this equation, low frequencies (for which s_k is not very different from s_{k-1}) are de-emphasized, whereas high frequencies are emphasized, improving the signal-to-noise ratio.

The next step is to divide the word into short snippets, a procedure referred to as *frame blocking*. Here, snippets of duration τ are extracted, with consecutive snippets shifted by $\delta\tau$. Note that $\delta\tau$ is typically smaller than τ, so that adjacent frames partially overlap. Once the frames have been generated, each frame is subjected to *windowing*, a procedure aimed at reducing discontinuities at the beginning and end of each frame. Thus, for each frame, the n samples are modified as

$$s_k \leftarrow s_k v_k, \tag{2}$$

where the (Hamming) windowing function takes the form

$$v_k = (1-\alpha) - \alpha\cos\frac{2\pi k}{n}, \tag{3}$$

where α is yet another parameter.

B. Feature extraction

Once the word has been preprocessed as described above, resulting in a set of frames, sound features are computed for each frame. Here, the sound features used have been (i) the autocorrelation coefficients, (ii) the linear predictive coding (LPC) coefficients, (iii) the cepstral coefficients, and (iv) the relative number of zero crossings. The autocorrelation coefficients are defined as

$$a_i = \sum_{k=1}^{n-i} \frac{(s_k-\overline{s})(s_{k+i}-\overline{s})}{\sigma^2}, \tag{4}$$

where $\overline{s}$ is the average of the samples and σ^2 is their variance. The number of extracted autocorrelation coefficients (i.e., the number of values of i used, starting from $i=1$) is referred to as the autocorrelation order.

The LPC and cepstral coefficients [10], which both are representations of the spectral envelope of a sound frame, have been used frequently in speech recognition [8]. The LPC coefficients l_i provide the best possible linear approximation of the sound, i.e., an approximation ($\hat{s}_k$) of sample s_k

$$\hat{s}_k = \sum_{i=1}^{p} l_i s_{k-i}, \tag{5}$$

for which the error $s_k - \hat{s}_k$ is minimal in the least square sense, where p is the LPC order, i.e., the number of extracted LPC coefficients.

Provided that the sound frame is quasi-stationary, which is often (almost) the case if the frame duration is set to a suitable value, the LPC coefficients provide an accurate compressed representation of the sound frame. The LPC coefficients can be derived efficiently from the autocorrelation coefficients, a procedure that will not be detailed here (see, for example, [8]). Once the LPC coefficients have been obtained, one can also compute the cepstral coefficients. These coefficients can be derived from the LPC coefficients using (non-linear) recursion. The detailed procedure can be found in [11]. The number of cepstral coefficients extracted is referred to as the cepstral order. Finally, the number of zero crossings n_zc is computed as the number of samples such that either the product $s_k s_{k-1} < 0$ or $s_k s_{k-2} < 0$ (if $s_{k-1} = 0$). Then, the *relative number of*

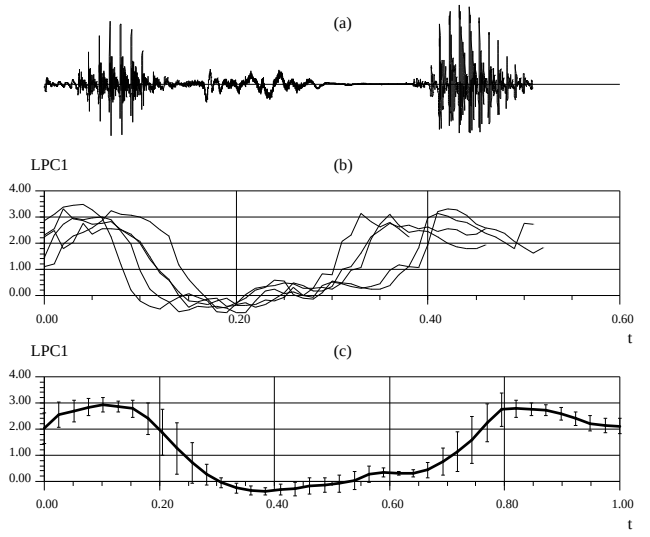

Fig. 1. An illustration of the averaging procedure (for a single feature, namely the first LPC coefficient, denoted LPC1) used when generating the template time series for a given word. Panel (a) shows one instance of the Swedish word *nästa* (meaning next). Panel (b) shows the original LPC1 time series obtained from five instances of the word *nästa*. As can be seen, the duration of the words, and therefore the number of feature points, varies a bit between instances. In panel (c), the time axes (for 10 instances of the word *nästa*) have been normalized, and the resulting series have been resampled, generating (in this case) 40 samples for each series, from which the average LPC1 time series, shown in panel (c), can be generated. This panel also shows the standard deviation (over the ten instances) for each sample.

zero crossings is formed by dividing $n_{\rm zc}$ by the number of samples (n) in the frame. With the procedure just described, several feature time series (with one set of features for each frame) are obtained for each word.

C. Speech recognition

Here, the SR (or, rather, word recognition) is based on a direct comparison between (a) feature time series stored for template words and (b) feature time series obtained from the spoken word. Thus, when generating the speech recognizer, a set of recorded template words is used. For each of the $n_{\rm w}$ words that the recognizer is supposed to cope with, $n_{\rm i}$ instances are recorded. The word instances are then preprocessed and feature time series are extracted, as described above.

Now, in order to generate a template time series for a given word, an average should be formed over the time series obtained for the $n_{\rm i}$ instances of the word in question. However, with a constant frame duration (τ) and a constant frame shift ($\Delta\tau$), the number of frames and, therefore, the number of elements in the time series, will depend on the duration of the spoken word instance. Thus, before forming the average, rather than using DTW to align the series, the time axes of all feature time series (for each instance) are simply linearly normalized to the range $[0, 1]$ and the resulting series are then re-sampled at equidistant (relative) times, resulting in an equal number of feature values for each feature time series and for each instance. Next, averages are formed over the $n_{\rm i}$ instances, which is straightforward since, after time normalization and re-sampling, all feature time series contain the same number of equidistant points. The average time series (one for each feature) are then stored in the speech recognizer. The process is repeated until all $n_{\rm w}$ words have been processed. The procedure is illustrated in Figure 1. For clarity, only five instances have been visualized in the middle panel, but the bottom panel is based on ten instances.

During speech recognition, the word to be recognized is subjected to the three steps of preprocessing, feature extraction, and resampling described above. Let F_{ijk} denote the $k^{\rm th}$ point of the $j^{\rm th}$ feature of stored word i (in the speech recognizer), and let φ_{jk} denote the $k^{\rm th}$ point of the $j^{\rm th}$ feature of the word to be recognized. The distance measure d_i is then computed as

$$d_i = \frac{1}{n_{\rm u} n_{\rm s}} \sum_{j=1}^{n_{\rm f}} w_j \sum_{k=1}^{n_{\rm s}} \left[\kappa_{jk} \left(F_{ijk} - \varphi_{jk}\right)\right]^2 . \qquad (6)$$

where $w_j \geq 0$ are feature weights. The inner sum (k) covers the number of time series points, or samples, (denoted $n_{\rm s}$) for each feature. The outer sum runs over the number of features ($n_{\rm f}$). $n_{\rm u}$ is the number of features for which w_j is different from 0. Thus, if $w_j > 0\ \forall j$, $n_{\rm u}$ is equal to $n_{\rm f}$. However, as shown below, faster SR performance can be obtained if some weights are set to 0. The κ_{jk} are scale factors (see below) that, in the basic distance measure (BDM), all take the value 1.

This distance measure is formed for each of the $n_{\rm w}$ stored words, and the index $i_{\rm r}$ of the word suggested by the speech recognizer is taken as

$$i_{\rm r} = {\rm argmin}_i d_i, \qquad (7)$$

if $d_{i_{\rm r}}$ is smaller than a threshold T. If not, the speech recognizer does not suggest any word. In order to simplify the comparison of results obtained in different runs, the threshold T was set to 1 for all runs. Note that, since the weights w_j are allowed to vary freely, in a wide range, it implies no restriction to set the threshold to a fixed value.

If the κ_{jk} are all equal to 1, all points along *a given feature* are weighted equally. However, as is evident from the bottom panel in Figure 1, the standard deviation (over the 10 different instances used when forming the average feature values) varies along the feature time series. For example, in that figure, one can see that feature points near the (normalized) time coordinate of around 0.20 have rather large standard deviation, whereas features points at time 0.60 have small standard deviation. One may thus argue that the latter points would perhaps be more useful in detecting the uttered word than feature points with larger standard deviation. Thus, a modified distance measure can be defined, henceforth referred to as the *standard deviation scaling* (SDS) distance measure[1], in which the κ_{jk} depend on the standard deviation for each feature

[1]Note that the approach introduced here is different from the *mean-variance normalization* (MVN) method [12] used in some SR systems. In MVN, the feature values are normalized using their estimated mean and variance over a sliding window. By contrast, in the approach considered here, the variance values over several stored instances are used for determining the κ_{jk} parameters.

 ISBN: 978-1-61208-224-0

point, from the stored words. For the SDS distance measure, the factors κ_{jk} are computed by first determining the values c_{jk} as

$$c_{jk} = \frac{1}{1 + \frac{\sigma_{jk}}{|\Phi_j|}} \tag{8}$$

where σ_{jk} is the standard deviation of feature point k along the time series for feature j, and $\overline{|\Phi_j|}$ is a normalization factor computed as the average modulus of the feature values along the series in question. The normalization factor is needed since different features have very different ranges. The modulus is introduced since many features have an average value near zero. Once the c_{jk} have been found, the κ_{jk} are computed as

$$\kappa_{jk} = n_{\rm s} \frac{c_{jk}}{\sum_{k=1}^{n_{\rm s}} c_{jk}}. \tag{9}$$

This slightly cumbersome procedure guarantees that the sum (over k) of κ_{jk} equals $n_{\rm s}$, making comparisons possible between runs that use the BDM (all κ_{jk} equal to 1) and runs that use the SDS distance measure.

D. optimization

Clearly, the distance measure depends on the weights w_j assigned for the different features. The simple choice of setting all weights to 1 is by no means optimal, since some features are more discriminative than others. Thus, an optimization procedure has been applied, during which the feature weights were optimized (for maximum SR performance). Note that the optimization did not involve the preprocessing, feature extraction, and resampling steps that, for a given word database, could thus be carried out once and for all.

For the purpose of optimization, a fairly standard GA [13] has been used, in which the feature weights are encoded in chromosomes (strings of floating-point numbers), and where the formation of new chromosomes takes place using standard tournament selection, single-point crossover, and creep mutations.

Three data sets were used in the GA runs: A training set, a validation set, and a test set. The results obtained over the training set were used as feedback to the GA, whereas the results obtained for the validation set, which were not provided to the GA, were used for determining when to stop the run in order to avoid overfitting. Once a run had been completed, the results obtained over the previously unused test set were taken as the true performance of the speech recognizer.

During the GA runs, the fitness Γ of an individual, i.e., a speech recognizer (with weights decoded from a chromosome), was computed as

$$\Gamma = \sum_{j=1}^{n_{\rm tr}} \gamma_j, \tag{10}$$

where the sum runs over all $n_{\rm tr}$ words in the training data set and γ_j is defined as

$$\gamma_j = \begin{cases} 1 + a(T-d) & \text{for correct identification,} \\ -b - a(T-d) & \text{for incorrect identification,} \\ a(T-d) & \text{if no word was recognized,} \end{cases} \tag{11}$$

TABLE I
THE PARAMETERS USED IN THE SPEECH RECOGNIZER.

Parameter	Value
Sound extraction threshold (t_p)	300
Sound extraction moving average length (μ)	10
Pre-emphasis parameter (c)	0.9373
Frame duration (τ)	0.030 (s)
Frame shift ($\Delta\tau$)	0.010 (s)
Hamming window parameter (α)	0.46
Autocorrelation order	8
LPC order	8
Cepstral order	12
Number of samples per feature ($n_{\rm s}$)	40

where d is the distance obtained for the word suggested by the speech recognizer, i.e., the distance d_i corresponding to index $i_{\rm r}$ in Equation (7). a and b are parameters, here set to 0.05 (a) and 0.50 (b). Thus, if $d \leq T$, a contribution (γ_j) slightly larger than 1 is given if the word was correctly identified, whereas a negative contribution is given if the wrong word was identified. Finally, if no word was recognized ($d > T$), a small negative contribution is given. With this fitness measure, the GA will attempt to find weights (see Equation (6)) that maximize the fraction of correctly identified words, as the prime objective, and also maximize the quantity $T - d$ for those words, as the second objective. Ideally, of course, $T - d$ should be as large as possible (for correctly identified words) since the risk of misidentification is then reduced for other instances of the word in question.

In some runs, described in Section III below, an effort was made to minimize the number of features used, by setting some weights to zero. In that case, the fitness Γ was multiplied by a penalty factor p defined as

$$p = 1 - \epsilon \frac{n_u}{n_f}, \tag{12}$$

where $\epsilon \ll 1$ is a positive constant. With this modification, the optimization procedure will favor speech recognizers using as few weights (and, therefore, features) as possible.

III. RESULTS

The three data sets (training, validation, and test) each contained 10 instances of 10 different words, i.e., a total of 100 sounds in each set. As the intended application is an interactive system for accessing online news (which will then be read by the agent or robot, typically as an aid to a visually impaired elderly person), the (minimal) vocabulary consisted of the ten Swedish words *ja* (yes) *nej* (no), *läs* (read), *åter* (return), *nästa* (next), *avsluta* (cancel or finish), *inrikes* (domestic (news)), *utrikes* (international), *ekonomi* (economy), and *sport* (sport, same as in English, but with different pronunciation). All sounds were sampled at 16 kHz.

After extensive testing, involving (short) GA runs for each parameter setting, the parameters used for preprocessing, feature extraction, and resampling were chosen as in Table I. As can be seen from the table, the total number of feature time series (for each word) was equal to $n_{\rm f} = 8 + 8 + 12 + 1 = 29$, including also the relative number of zero crossings.

 ISBN: 978-1-61208-224-0

TABLE II
OPTIMIZATION RESULTS: COLUMN 1 INDICATES THE RECOGNIZER TYPE, WHICH IS DETERMINED BY THE PARAMETERS SHOWN IN COLUMNS 2 AND 3. DM = DISTANCE MEASURE, BDM = BASIC DISTANCE MEASURE, SDS = STANDARD DEVIATION SCALING. COLUMNS 4 TO 6 SHOW THE FRACTION OF CORRECTLY RECOGNIZED WORDS FOR THE TRAINING, VALIDATION, AND TEST SETS, RESPECTIVELY. COLUMN 7 SHOWS THE AVERAGE DISTANCE VALUE (SEE EQUATION (6)) FOR THE CORRECTLY CLASSIFIED WORDS IN THE TEST SET.

Type	n_{u}	DM	Training	Validation	Test	$\overline{d}_{\min}$
1	29 (all)	BDM	1.00	0.99	0.99	0.3209
2	5	BDM	1.00	1.00	0.98	0.1466
3	29 (all)	SDS	0.98	0.91	0.91	—
4	8	SDS	0.98	0.93	0.93	—

Next, several long GA runs were carried out. The population size was set to 30, the tournament selection parameter to 0.75, the crossover probability to 0.80, and the mutation probability to $1/n_{\mathrm{p}}$, where n_p is the number of optimizable parameters (i.e., the weights w_j). In runs using the fitness measure from Equation (10), the weights w_j took values in the range $[0, 5]$. In runs using the fitness measure from Equation (12), the weight range was the same but, in addition, weights could be set to zero (exactly) with a probability of around 0.50.

The results of the best run (i.e., the run with highest validation fitness) for each of four different speech recognizer types are given in Table II. In runs with types 1 and 3, the standard fitness measure (without weight penalty) was used whereas for runs with types 2 and 4, the weight penalty was included in the fitness measure; see Equation (12). In types 1 and 2 the BDM was used, whereas types 3 and 4 used the SDS distance measure. Rather than the fitness values, the table shows the (more relevant) fraction of correctly recognized words for the training, validation, and test sets, respectively, for the best recognizer found of each type. In the rightmost column, the average values of $d_{\min} \equiv d_{i_r}$ (i.e., the distance measure for the recognized word) are shown for the correctly classified words for types 1 and 2, but not for types 3 and 4, since their test performance was too low for the average $d_{\min}$ values to be meaningful.

In the speech recognizer with highest validation fitness (type 2 in Table II), which reached a perfect result on the training and validation sets, and nearly perfect performance on the test set, the five features used were only cepstral coefficients, namely coefficients number 3,7,8,11, and 12. Even though the type 1 recognizer reached a slightly better result on the test set, one can argue that the type 2 recognizer is better, since it has a smaller average $d_{\min}$ value.

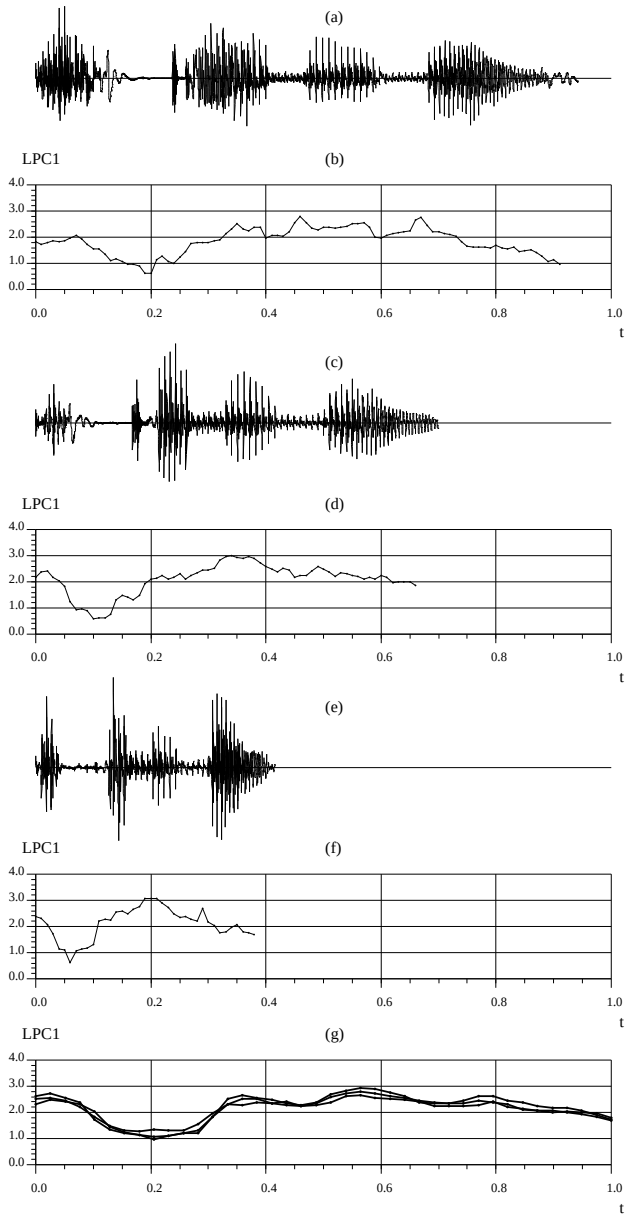

Fig. 2. An illustration of the fact that linear time normalization generates similar (average) feature time series, shown in Panel (g), regardless of the speaking speed. Panels (a)-(f) show the sound samples and the first LPC coefficient for three utterances of the word *ekonomi*, using slow, standard, and fast speaking, respectively. See the main text for a full description of the figure.

IV. DISCUSSION

The results in Table II show that the proposed method, with the basic distance measure (all κ_{jk} equal to 1), works well, and that some weights can be set to zero, without significant loss in performance. This is important, since the time needed to recognize a word may be crucial if larger vocabularies are used. The time needed for recognition consists of a part (the feature extraction) that, for a given set of features, depends only on the number of samples in the recognized word and a part (the computation of d_i) that is linear in the number of stored (recognizable) words (n_{w}). The first part is dominated by the time needed to compute autocorrelations, but can be somewhat reduced if only some LPC and cepstral coefficients are needed. In the setup used here (running on a 2.67 GHz core i7 processor) using a typical word size of around 10,000 samples, the constant part of the recognition time is around 0.0672 s, and the linear part is equal to $1.11 \times 10^{-4} n_{\mathrm{w}}$ s, if all weights are non-zero (type 1 in Table II) and $0.224 \times 10^{-4} n_{\mathrm{w}}$ s

for the best recognizer found, i.e., type 2 in Table II. Thus, if a maximum recognition time of, say, 0.20 s is allowed for real-time performance, the number of words that can be stored increases from around 1,200 (for type 1) to around 5,900 (for type 2).

The results also show that SDS had a detrimental effect on SR performance. A likely reason for the reduced performance is the fact that the standard deviation estimates are not very accurate since they are based on only 10 instances. Of course, the number of instances could be increased, but that would make the process of generating the speech recognizer quite time-consuming, at least for larger vocabularies than in this example. On the other hand, using SDS is not really needed, since the BDM reaches near-perfect performance.

As mentioned in Section I, direct comparison of time series in SR is usually carried out using DTW rather than (linear) time normalization followed by resampling as used here. However, the motivation for using the DTW approach is not very strong. First of all, Ratanamahatana and Keogh [14] have noted that time series of different length can simply be renormalized to equal length without loss in recognition performance. Furthermore, Boulgouris *et al.* [15] concluded that linear time normalization in fact *outperformed* DTW in the context of gait identification.

Our results confirm these findings, and a clear illustration of this fact is given in Figure 2. Here, the word *ekonomi* (economy) was uttered ten times slowly (average duration: 0.907 s), ten times with normal speed (average duration: 0.634 s), and ten times quickly (average duration: 0.433 s). The top six panels of the figure show one instance of the slow, normal, and fast utterances, along with the corresponding original time series for the first LPC coefficient (LPC1). The bottom panel shows the *average* feature time series over the ten instances of slow, normal, and fast readings, respectively, after linear time normalization and resampling. Despite the very different reading speeds, the feature values are very similar. In fact, the differences between the three curves are of the same order of magnitude as the standard deviations (not shown) over the ten instances for each curve separately. In all three cases (slow, normal, and fast), the best speech recognizers, namely types 1 and 2 in Table II, correctly identified all ten instances.

The work presented here represents the initial development stage of an intelligent agent with Swedish speech recognition, and there are some obvious limitations, namely that (i) a rather limited vocabulary was used, (ii) the sounds were recorded by a single speaker, and (iii) the system was trained using only the words in the vocabulary, i.e., no false positives were included. However, it should be noted that even with the present training setup, the speech recognizer can avoid incorrect detection of unknown words, namely in cases where the uttered word is sufficiently different from the stored words so that the resulting minimum distance exceeds the threshold T. As for speaker independence, even though the speech recognizer was trained using a single voice, some initial tests with other speakers showed promising results, which, however, will be followed up as described below.

V. Conclusion and further work

To conclude, it has been shown that linear time normalization followed by feature matching provides robust speech recognition, and that, with optimization, the recognition speed can be strongly improved (especially important for large vocabularies), by using only a subset of the available features.

Even though the size of the vocabulary used here is sufficient for the application at hand (i.e., an intelligent news reader agent; see Section I), and the intended use is as a companion to a single elderly person, the next step will be to increase the size of the vocabulary and, in doing so, use words spoken by different people. In addition, the issue of training with false positives present will be investigated. Another obvious topic for further work would be to extend the method in order to handle not only IWR but CSR as well.

References

[1] Eurostat news release 80/2011, "EU27 population is expected to peak by around 2040". Accessed Nov. 29, 2011. [Online]. Available: http://epp.eurostat.ec.europa.eu/cache/ITY_PUBLIC/3-08062011-BP/EN/3-08062011-BP-EN.PDF

[2] J. Broekens, M. Heerink, and H. Rosendal, "Assistive social robots in elderly care: a review," *Gerontechnology*, vol. 8, no. 2, pp. 94–103, 2009.

[3] M. Chan, D. Estève, C. Escriba, and E. Campo, "A review of smart homes - present state and future challenges," *Computer Methods and Programs in Biomedicine*, vol. 91, pp. 55–81, 2008.

[4] K. Wada and T. Shibata, "Living with seal robots - its sociopsychological and physiological influences on the elderly at a care house," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 972–980, 2007.

[5] C. Granata, M. Chetouani, A. Tapus, P. Bidaud, and Dupourqué, "Voice and graphical-based interfaces for interaction with a robot dedicated to elderly and people with cognitive disorders," in *Proceedings of the 19th IEEE International Symposium on Robot and Human Interactive Communication*, 2010, pp. 785–790.

[6] A. Lipeika and J. Lipeikienė, "On the use of the formant features in the dynamic time warping based recognition of isolated words," *Informatica*, vol. 19, no. 2, pp. 213–226, 2008.

[7] F. Rosdi and R. Ainon, "Isolated malay speech recognition using hidden markov models," in *Proceedings of the International Conference on Computer and Communication Engineering*, 2008, pp. 721–725.

[8] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Prentice Hall, 1993.

[9] P. Wong, O. Au, J. Wong, and W. Lau, "Reducing computational complexity of dynamic time warping based isolated word recognition with time scale modification," in *Proceedings of the Fourth International Conference on Signal Processing*, 1998, pp. 722–725.

[10] J. Markel and A. Gray Jr., *Linear prediction of speech*. Springer Verlag, 1976.

[11] G. Antoniol, V. Rollo, and G. Venturi, "Linear predictive coding and cepstrum coefficients for mining time invariant information from software repositories," in *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, 2005, pp. 1–5.

[12] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Communication*, vol. 25, pp. 133–147, 1998.

[13] J. H. Holland, *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[14] C. Ratanamahatana and E. Keogh, "Three myths about dynamics time warping data mining," in *Proceedings of SIAM International Conference on Data Mining*, 2005, pp. 506–510.

[15] N. Boulgouris, K. Plataniotis, and D. Hatzinakos, "Gait recognition using linear time normalization," *Pattern Recognition*, vol. 39, pp. 969–979, 2006.

 ISBN: 978-1-61208-224-0