Chapter 5

Improvement Techniques for Automatic Speech Recognition

Santiago Omar Caballero Morales

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/49969

1. Introduction

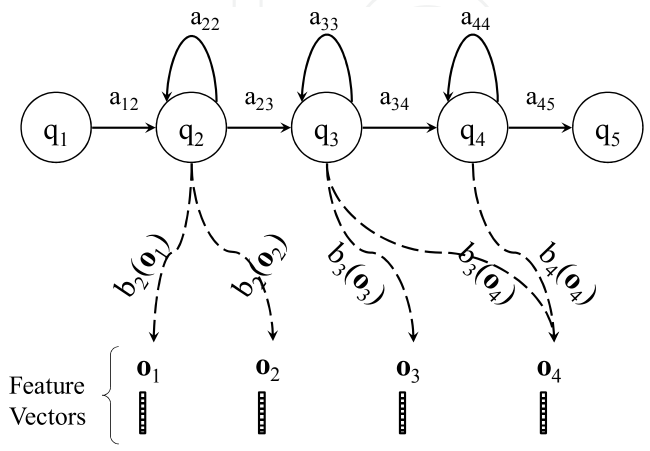
Research on spoken language technology has led to the development of Automatic Speech Recognition (ASR), Text-To-Speech (TTS) synthesis, and dialogue systems. These systems are now used for different applications such as in mobile telephones for voice dialing, GPS navigation, information retrieval, dictation, translation, and assistance for handicapped people.

To perform Automatic Speech Recognition (ASR) there are different techniques such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Weighted Finite State Transducers (WFSTs), and Hidden Markov Models (HMMs). In the case of HMMs there are algorithms such as Viterbi, Baum - Welch / Forward - Backward, that adjust their parameters and the decoding/recognition process itself. However these algorithms do not guarantee optimization of these parameters as the recognition process is stochastic.

Main challenges arise when the structures used to describe the stochastic process (i.e., a three-state left-to-right HMM) are not enough to model the acoustic features of the speech. Also, when training data to build robust ASR systems is sparse. In practice, both situations are met, which leads to decrease in rates of ASR accuracy. Thus, research has focused on the development of techniques to overcome these situations and thus, to improve ASR performance. In the fields of heuristic optimization, data analysis, and finite state automata, diverse techniques have been proposed for this purpose. In this chapter, the theoretical bases and application details of these techniques are presented and discussed.

Initially, the optimization approach is reviewed in Section 2, where the application of heuristic methods as Genetic Algorithms and Tabu Search for structure and parameter optimization of HMMs is presented. This continues in Section 3, where the application of WFSTs and discrete HMMs for statistical error modelling of an ASR system's response is presented. This approach is proposed as a corrective method to improve ASR performance. Also, the use of a data analysis algorithm as Non-negative Matrix Factorization (NMF) is presented as a mean to improve the information obtained from sparse data. Then, case studies where these techniques are applied, and statistically significant improvements on ASR performance is achieved, are presented and discussed in Section 4. Finally, in Section 5 the main conclusions and observations about the techniques and their performance on ASR development are presented.

2. Techniques for optimization of parameters



Matrix of Transition Probabilities

$$A=\begin{array}{cc}\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} & \begin{array}{c}1\\2\\3\\4\\5\end{array} \\ \begin{array}{ccccc}1 & 2 & 3 & 4 & 5\end{array} & \end{array}$$

Emission Probabilities (Observation Likelihoods)

$$B=\left\{ b_j(\mathbf{o}_t)=\sum_{k=1}^{K} C_{jk} N(\mathbf{o}_t, \boldsymbol{\mu}_{jk}, \Sigma_{jk}) \right\}$$

Figure 1. Parameters of an HMM.

In Figure 1 the standard structure and parameters of an HMM for acoustic modelling at the phonetic level are shown [10]. The notation $\lambda = (A, B, \pi)$ is used to indicate these parameters, where π is an initial state distribution [1]. There are three main problems associated with HMMs, and thus, with the estimation of these parameters:

- The evaluation problem. Given the parameters of a model (λ), estimate the probability of a particular observation sequence ($Pr(\boldsymbol{O}|\lambda)$).
- The learning problem. Given a sequence of observations $\boldsymbol{o}_t$ from a training set, estimate/adjust the transition (A) and the emission (B) probabilities of an HMM to describe the data more accurately.
- The decoding problem. Given the parameters of the model, find the most likely sequence of hidden states $Q^* = \{q_1, q_2, ..., q_n\}$ that could have generated a given output sequence $\boldsymbol{O} = \{\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_t\}$.

Standard algorithms such as Viterbi (for decoding) and Baum-Welch (learning) are widely used for these problems [1]. However, heuristics such as Tabu Search (TS) and Genetic Algorithms (GA) have been proposed to further improve the performance of HMMs and the parameters estimated by Viterbi/Baum-Welch.

A GA is a search heuristic that mimics the process of natural evolution and generates useful solutions to optimization problems [2]. In Figure 2 the general diagram of a GA used for HMM optimization is show, where the solutions for an optimization problem receive values based on their quality or "fitness", which determine their opportunities for reproduction. It is expected that parent solutions of very good quality will produce offsprings (by means of reproduction operators such as crossover or mutation) with similar or better characteristics, improving their fitness after some generations. Hence, fitness evaluation is a mechanism used to determine the confidence level of the optimized solutions to the problem [9]. In Figure 3 a general "chromorome" representation of an individual of the GA population, or parent solution, is shown. This array contains the elements of an HMM (presented in Figure 1), which can be coded into binary format to perform reproduction of solutions.

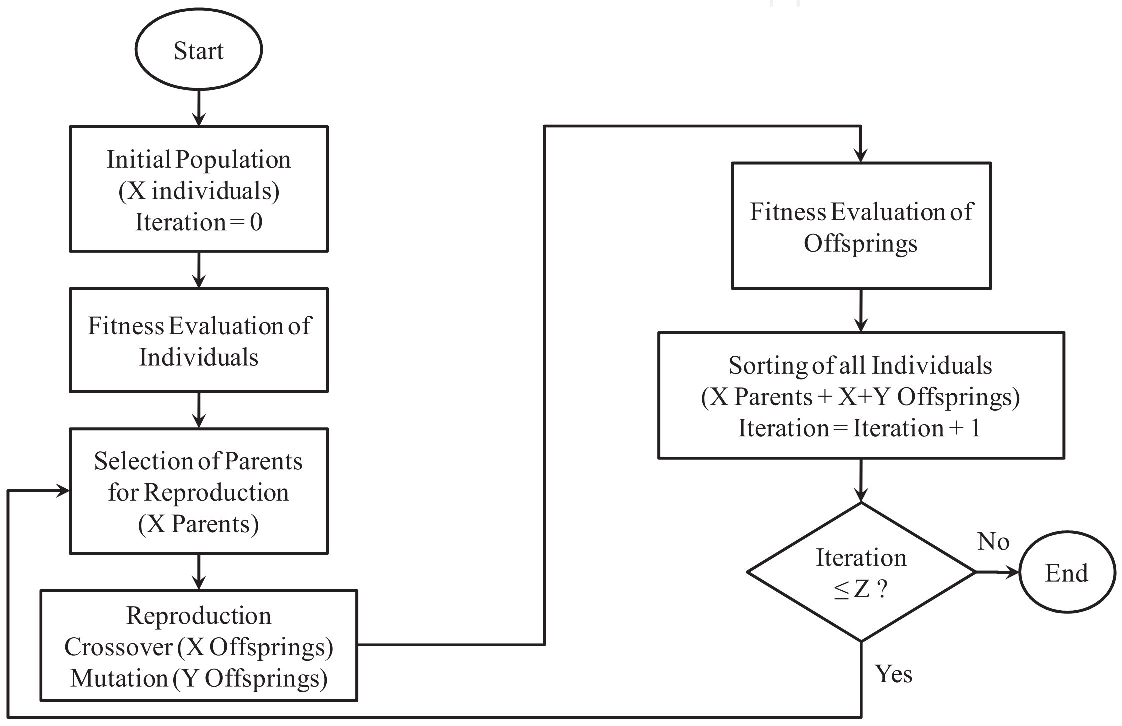

Start
Initial Population (X individuals) Iteration = 0
Fitness Evaluation of Individuals
Selection of Parents for Reproduction (X Parents)
Reproduction Crossover (X Offsprings) Mutation (Y Offsprings)
Fitness Evaluation of Offsprings
Sorting of all Individuals (X Parents + X+Y Offsprings) Iteration = Iteration + 1
Iteration ≤ Z ?
No
End
Yes

Figure 2. General diagram of a GA for optimization of HMMs.

		Transition Matrix					Observation Probabilities					Initial State Distribution				
N	M	a_{11}	…	…	…	a_{NN}	b_{11}	…	…	…	b_{NM}	π_1	…	…	…	π_N

Figure 3. Chromosome array of the elements of an HMM, where *N* defines the number of states, and *M* the number of mixture components of the observation probabilities' distributions.

In [3], GA optimization was performed for the observation probabilities and transition states for word HMMs, while in [8] finding an optimal structure was the objective. In [8], the fitness of individuals (HMM structures) was measured considering (1) the log-likelihood sum of the HMM calculated by the Viterbi algorithm over a training set of speech data, and (2) a penalization based on the error rate obtained with the HMM when performing Viterbi over a sub-set of test data. [9] used a GA to optimize the number of states in the HMM and its parameters for web information extraction, obtaining important increases in precision rates when compared with Baum-Welch training. Fitness was measured on the likelihood $(Pr(\boldsymbol{O}|\lambda))$ over a training set.

On the other hand, TS is a metaheuristic that can guide a heuristic algorithm from a local search to a global space to find solutions beyond the local optimality [11]. It can avoid loops in the search process by restricting certain "moves" that would make the algorithm to revisit a previously explored space of solutions. These moves are kept hidden or reserved (are being kept "Tabu") in a temporal memory (a Tabu List) which can be updated with new moves or released with different criteria. While in a GA the diversification of the search process is performed by the reproduction operators, in TS this is performed by "moves" which consist of perturbances (changes) in the parameter values of an initial solution (i.e., observation or transition probabilities). These changes can be defined by a function, or by adding or substracting small randomly generated quantities to the initial solution's values.

This approach was explored by [27] for HMM optimization. As in other studies, the log probability indicating the HMM likelihood over a training set was used to measure the fitness of a solution. The "moves" consisted in adding randomly generated values to each HMM's parameters. In the next section, improvement techniques based on statistical error modelling of phoneme confusions are presented and discussed.

3. Statistical error modelling techniques

Modelling of phoneme confusions has been explored to estimate error patterns in the articulation and automatic recognition of speech. The statistical modelling of this information has led to achieve improvements in the performance of ASR systems. In [17] a word confusion-matrix was used to predict the performance of a speaker-dependent ASR system. This allowed the removal of words more likely to be confused from a selected vocabulary, and to form a better set of command-words for control applications.

A similar "vocabulary design" approach was also used in [18], where a phoneme confusion - matrix was used to incorporate alternative transcriptions of a word in the dictionary, and thus, reduce its confusion probability with other words. A chinese - character confusion - matrix was used in [23] to get more accurate "candidates" to increase recognition of chinese scripts. In [20], the modelling of phoneme confusions was performed to correct phoneme recognition errors. In this section, three main techniques to perform statistical modelling of phoneme-confusions patterns are presented:

- Non-negative Matrix Factorization, NMF (Section 3.1.1). Application case in Section 4.2 (with Metamodels).
- Metamodels (Section 3.2). Application case in Section 4.1 (with GA), 4.2 (with NMF), and 4.3 (with WFSTs).
- Weighted Finite State Transducers, WFSTs (Section 3.3). Application case in Section 4.3 (with Metamodels).

3.1. Phoneme confusion-matrix as resource for error modelling

In the field of artificial intelligence, a confusion-matrix is a visualization tool typically used in supervised learning. Each column of the matrix represents the instances in a recognized class, while each row represents the instances in an actual class [15]. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (e.g., commonly mislabelling or classifying one as another).

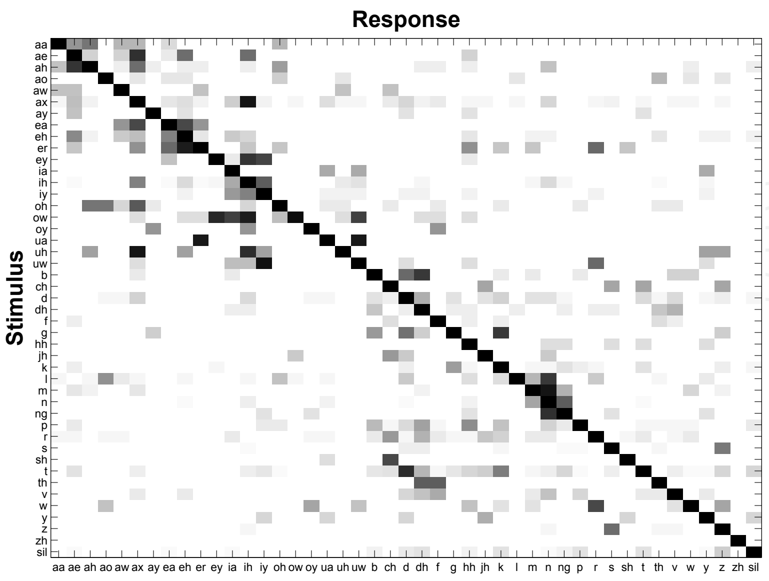


Figure 4. Example of a phoneme confusion-matrix.

As shown in Figure 4, in a phoneme confusion-matrix, rows represent the phonemes intended or uttered by the speaker (*stimulus phonemes*), and the columns represent the decoded phonemes given by the ASR system (*response*)[1]. The classification of phonemes to estimate a phoneme confusion - matrix is performed by the alignment of two phoneme strings (or sequences):

- P, the reference (correct) phoneme transcription of the sequence of words W uttered by a speaker.
- $\tilde{P}^*$, the sequence of phonemes decoded by the ASR system.

As $\tilde{P}^*$ is the system's output, it might contain several errors. Based on the classification performed by the aligner, these are identified as substitution (S), insertion (I), and deletion (D) errors. Thus, the performance of ASR systems is measured based on these errors, and two metrics are widely used for phoneme and word ASR performance:

$$Word\_Accuracy(WAcc) = \frac{N - D - S - I}{N}, \quad Word\_Error\_Rate(WER) = 1 - WAcc \quad (1)$$

where N is the number of elements (words or phonemes) in the reference string (P). Thus, the objective of the statistical modelling of the phoneme confusion-matrix is to estimate W from $\tilde{P}^*$. This can be accomplished by the following expression [22]:

$$W^* = \max_{P} \prod_{j}^{M} Pr(p_j) Pr(\tilde{p}_j^* | p_j) \quad (2)$$

[1] In this case, each class is a phoneme in the British-English BEEP pronunciation dictionary [13] which considers 45 phonemes (vowels and consonants).

where p_j is the j'th phoneme in the postulated phoneme sequence P, and $\tilde{p}_j^*$ the j'th phoneme in the decoded sequence $\tilde{P}^*$ (of length M). Equation 2 indicates that the most likely word sequence is the sequence that is most likely given the observed phoneme sequence from a speaker. The term $Pr(\tilde{p}_j^*|p_j)$ represents the probability that the phoneme $\tilde{p}_j^*$ is recognized when p_j is uttered, and is obtained from a speaker's confusion-matrix. This element is integrated into the recognition process as presented in Figure 5.

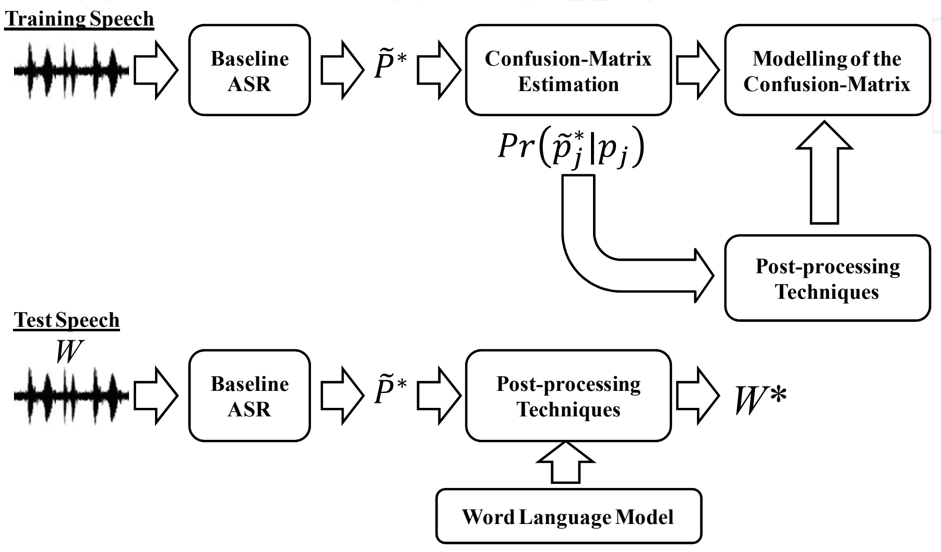


Figure 5. Training and testing process of error modelling techniques.

This information then can be modelled by post-processing techniques to improve the baseline ASR's output. Evaluation is performed when $\tilde{P}^*$ (which now is obtained from test speech) is decoded by using the "trained" techniques into sequences of words W^*. The correction process is done at the phonetic level, and by incorporating a word-language model a more accurate estimate of W is obtained. In the sections 3.2 and 3.3, the foundations of two post-processing techniques are explained.

3.1.1. Non-negative matrix factorization to improve phoneme confusion - matrix estimates

An important problem is observed when the data available for confusion-matrix estimation is small, which leads to poor estimates, and in practice, this is the normal situation. An approach presented by [28] and [30] made use of Non-negative Matrix Factorization (NMF) to find structure within confusion-matrices from a set of "training" speakers, which then could be used to make improved estimates for a "test" speaker given only a few samples from his/her speech. An advantage of this technique is that it was able to remove some of the noise present in the sparse estimates, while retaining the particular speaker's confusion-matrix patterns. This approach is reviewed in this section.

NMF is more suitable to estimate confusion-matrix probabilities (e.g., $Pr(\tilde{p}_i{}^*|p_i)$), as these are non-negative. This is a property not shared by similar methods as Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Note that, although there is no guarantee that the NMF estimates will be in the range [0,1], the normalizations required are less severe than those required if negative estimates are present.

NMF seeks to approximate an $n \times m$ non-negative matrix V by the product of two non-negative matrices W and H:

$$V \approx WH. \tag{3}$$

where W is a $n \times r$ matrix, H is a $r \times m$ matrix, and $r \leq min(n, m)$. When $r < min(n, m)$, the estimate of V, $\hat{V} = WH$, can be regarded as having being projected into and out of a lower-dimensional space r [6]. The columns of W are regarded as forming a set of (non-orthogonal) basis vectors that efficiently represent the structure of V, with the columns of H acting as weights for individual column vectors of V [28]. Estimation of V is accomplished by minimizing a distance function between WH and V, which is defined by the Frobenius norm [6].

In this case, for a speaker S_y, it is assumed that two types of confusion-matrices exist, (1) a **target** confusion-matrix, which is estimated using all the available utterances from that speaker and is designated as CM^y, and (2) **partial** confusion-matrices, defined as CM_U^y, which are estimated by using some U utterances. Thus, for NMF each column of V is a **target** confusion-matrix CM^x (written out column by column) from a training-set speaker S_x. To estimate a **target** confusion-matrix CM^y from a **partial** confusion-matrix CM_U^y of a "test" speaker S_y, CM_U^y is added as an extra column to V. When NMF is applied to V, the estimated confusion-matrix $\widehat{CM}^y$ is retrieved from $\hat{V}$ and is re-normalized so that its rows sum to 1.0. A weighted distance squared difference measure $D(CM^y, \widehat{CM}^y)$ can be used to assess the quality of the estimates of $\widehat{CM}^y$. The process, presented in Figure 6 is iterated until the obtained estimates converge.

If the data available from a speaker is very small, the **partial** confusion-matrix CM_U^y will be too sparse to make an improved estimate using NMF. Hence, CM_U^y can be smoothed by using a speaker-independent confusion-matrix $\overline{CM}$, which is well estimated from the training data. If the total number of non-zero elements in CM_U^y is less than a given threshold (TH), the row can be replaced by the equivalent row of $\overline{CM}$.

3.2. Metamodels

In practice, it is too restrictive to use only the confusion-matrix to model $Pr(\tilde{p}_j^*|p_j)$ as this cannot model insertions well. Instead, a Hidden Markov model (HMM) can be constructed for each of the phonemes in the phoneme inventory. These HMMs, termed as *metamodels* [22, 24], can be best understood by comparison with a "standard" acoustic HMM: a standard acoustic HMM estimates $Pr(O'|p_j)$, where O' is a subsequence of the complete sequence of observed acoustic vectors in the utterance, O, and p_j is a postulated phoneme in P. A metamodel estimates $Pr(\tilde{P}'|p_j)$, where $\tilde{P}'$ is a subsequence of the complete sequence of observed (decoded) phonemes in the utterance $\tilde{P}$.

The architecture of the metamodel of a phoneme is shown in Figure 7 [22, 24]. Each state of a metamodel has a discrete probability distribution over the symbols for the set of phonemes, plus an additional symbol labelled DELETION. The central state (2) of a metamodel for a certain phoneme models correct decodings, substitutions and deletions of this phoneme made by the phoneme recognizer. States 1 and 3 model (possibly multiple) insertions before and after the phoneme. If the metamodel were used as a generator, the output phone sequence produced could consist of, for example:

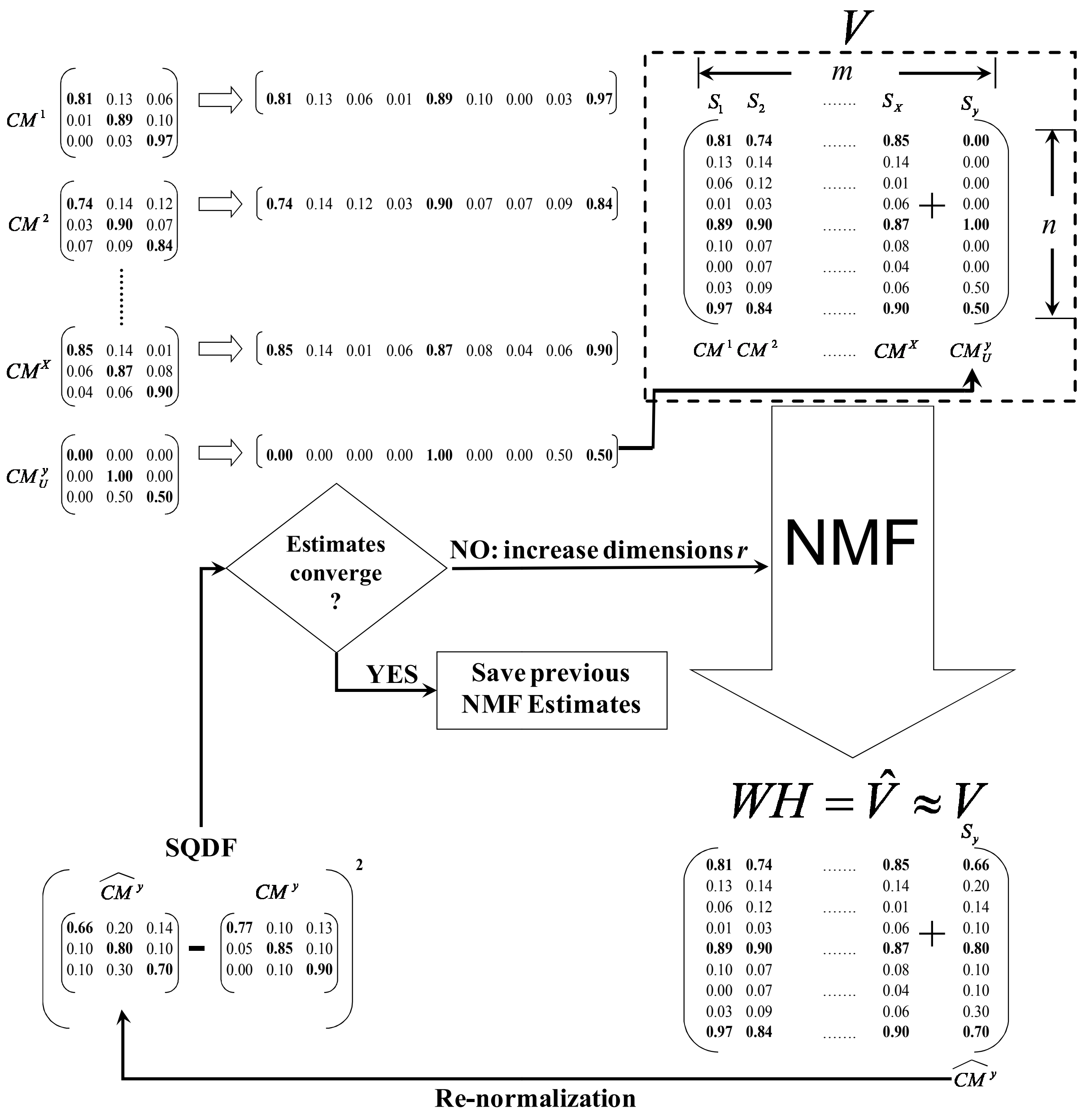


Figure 6. NMF Estimation Process.

- a single phone which has the same label as the metamodel (a correct decoding) or a different label (a substitution);
- a single phone labelled DELETION (a deletion);
- two or more phones (one or more insertions).

As an example of the operation of a metamodel, consider a hypothetical phoneme that is always decoded correctly without substitutions, deletions or insertions. In this case, the discrete distribution associated with the central state would consist of zeros except for the probability associated with the symbol for the phoneme itself, which would be 1.0. In addition, the transition probabilities a_{02} and a_{24} would be set to 1.0 so that no insertions could

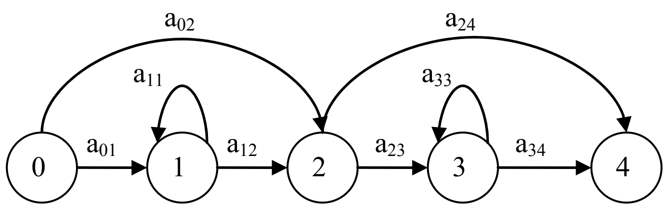


Figure 7. Metamodel of a phoneme.

be made. When used as a generator, this model can produce only one possible phoneme sequence: a single phoneme which has the same label as the metamodel.

The discrete probability distributions of each metamodel can be refined by using embedded re-estimation with the Baum-Welch Algorithm [10] over the $\{P, \tilde{P}^*\}$ pairs of all the utterances. When performing speech recognition, the language model is used to compile a "meta-recognizer" network, which is identical to the network used in a standard word recognizer except that the nodes of the network are the appropriate metamodels rather than the acoustic models used by the word recognizer. As shown in Figure 5, the output test phoneme sequence $\tilde{P}^*$ is passed to the meta-recognizer to produce a set of word hypotheses.

3.2.1. Extended metamodels

An extension to the original metamodel was presented by [29], noting that only one state is used to model multiple insertions in the original metamodel. Because of the first-order Markov property, this would not be able to model patterns within insertions. In order to cover a higher number of insertions, and identify their pattern of substitutions, the insertion-states were extended as shown in Figure 8.

Having information about the pattern of substitutions of an insertion can provide support to the modelling of the "insertion-context" of a phoneme. B_j represents the j-th *insertion-before* the phoneme, and A_k is the k-th *insertion-after* the phoneme. These $j = 1 \ldots J$ and $k = 1 \ldots K$ indexes identify the contexts of such insertions, where J and K represent the length of the contexts.

For the modelling of the insertions **"before"** a phoneme, consider the Figure 9(a). As the "insertion-context" is taken with reference of the central state (C), state B_1 models the first-order insertions, state B_2 the second-order insertions, and so until state B_J, where J is the length of the context. From left-to-right, state B_{J+1} is the "initial" state, and B_0 is the central state (as state 0 and state 2 in Figure 7).

While the probabilities within each state are estimated from the normalized confusion - matrices, the transition probabilities a between states are estimated from the non - normalized confusion - matrices. Figure 9(a) shows a general example of the counts (ocurrences of phonemes) within each state B_j, where $C(B_j)$ represents the total counts in that state. Note that $C(B_0) = C(C)$, and $C(B_{J+1}) = 0$, as the initial state has null observations (non-emitting state). ΔB_j represents the number of elements that do the *transition* to a particular state B_j, and is expressed as:

$$\Delta B_j = C(B_j) - C(B_{j+1}) \tag{4}$$

The transition probabilities a_B for the "insertion-before" states are then computed as:

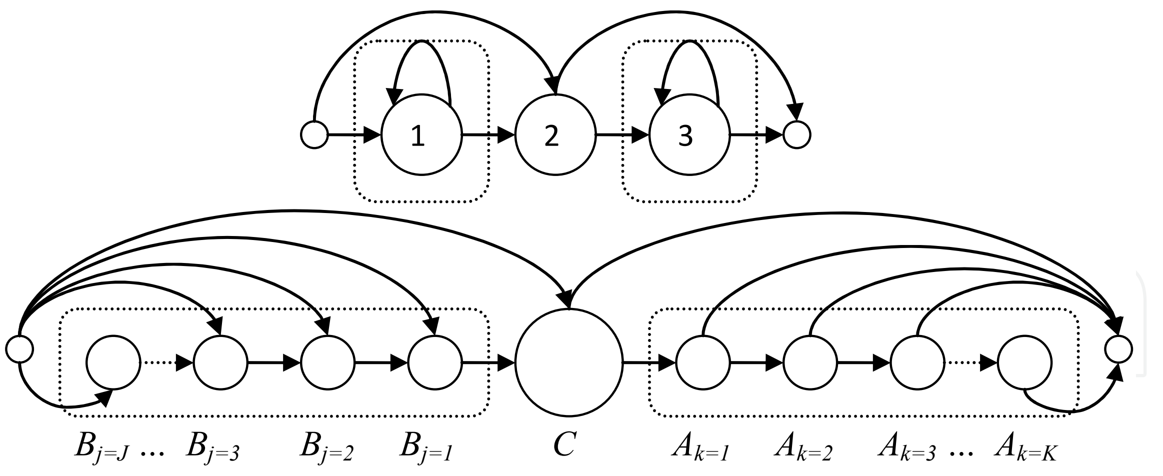


Figure 8. Extended metamodel of a phoneme. The states A_k and B_j represent the k-th insertion-after, and the j-th insertion-before, of a phoneme.

```
for $j$ = 0 to J do
    $a\_B_{J+1,j} = \frac{C(B_j)-C(B_{j+1})}{C(B_0)} = \frac{\Delta B_j}{C(B_0)}$
end for
```

The above algorithm also gives the transition probability from the initial state to the central state when $j = 0$ ($a\_B_{J+1,0}$). When a transition ends into an insertion state (B_j), the next transition must be to the preceding insertion state (B_{j-1}) because there is a dependency on their ocurrences, so:

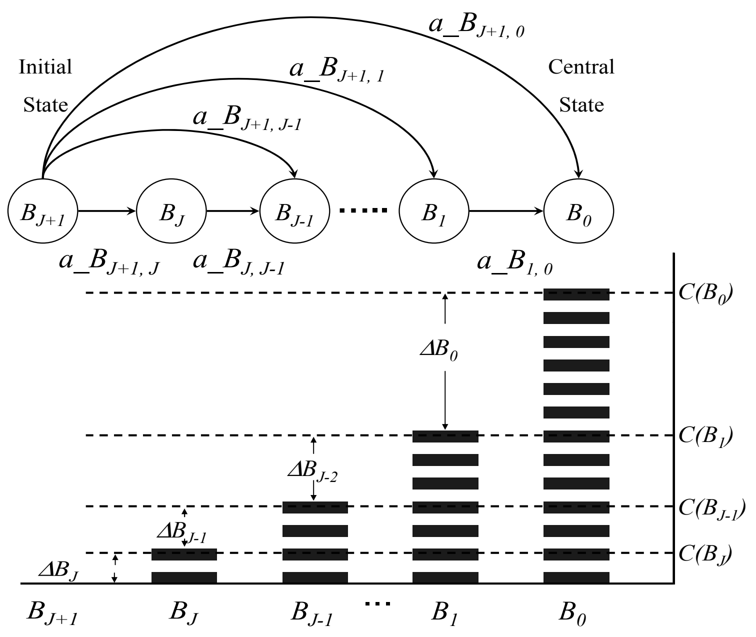
```
for $j$ = 0 to J-1 do
    $a\_B_{j+1,j} = 1$
end for
```

The modelling of the insertions **"after"** a phoneme is performed in a slightly different way as the transition sequences change. As shown in Figure 9(b), the "insertion-context" is taken with reference of the central state (C), where state A_1 models the first-order insertions, state A_2 the second-order insertions, and so until state A_K, where K is the length of the context. From left-to-right, state A_0 is the central state, and A_{K+1} is the final or end state (as state 2 and state 4 in Figure 7). $C(A_0) = C(B_0) = C(C)$, and $C(A_{K+1}) = 0$ as the final state is non-emitting.

When starting on the central state there must be a sequence for the next transitions, thus either going to the final state A_{K+1} or to the first-order insertion-after state A_1. If the above transition ended in A_1, then the next transition could be to the second-order insertion-after state A_2, or to the final state A_{K+1}, and so on. ΔA_k represents the number of elements or observations that move from an insertion-after state A_k to the final state A_{K+1} and is expressed as:

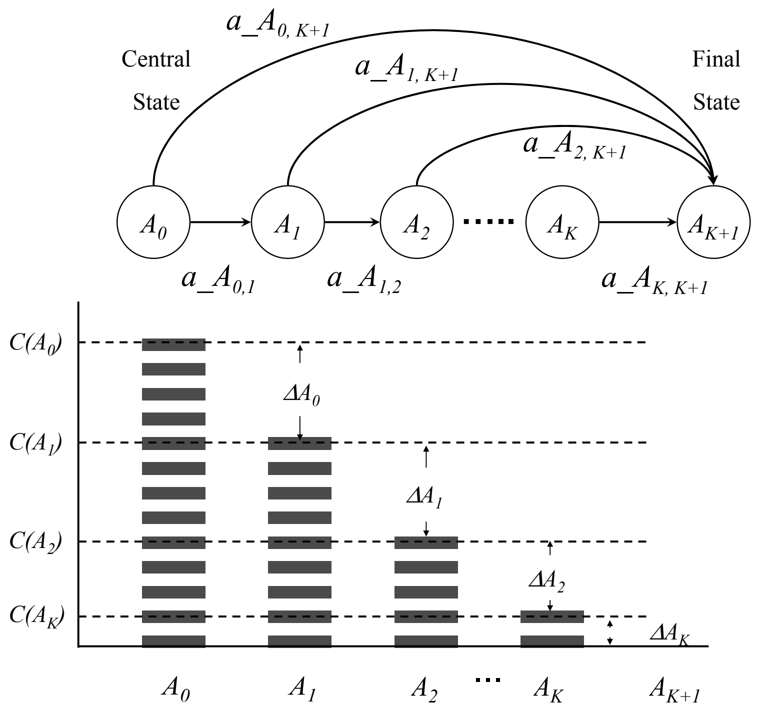
$$\Delta A_k = C(A_k) - C(A_{k+1}) \tag{5}$$

The remaining observations, $C(A_k)$-ΔA_k, do the transition to the state A_{k+1}, as the elements in A_{k+1} are dependent on the same number of elements in A_k. The transition probabilities a_A for each insertion-after state are computed as:



$a\_B_{J+1,0}$
Initial State
$a\_B_{J+1,1}$
Central State
$a\_B_{J+1,J-1}$
B_{J+1} B_J B_{J-1} B_1 B_0
$a\_B_{J+1,J}$ $a\_B_{J,J-1}$ $a\_B_{1,0}$
$C(B_0)$ $C(B_1)$ $C(B_{J-1})$ $C(B_J)$
ΔB_0 ΔB_{J-2} ΔB_{J-1} ΔB_J

(a) Insertions "before" a phoneme



$a\_A_{0,K+1}$
Central State
$a\_A_{1,K+1}$
Final State
$a\_A_{2,K+1}$
A_0 A_1 A_2 A_K A_{K+1}
$a\_A_{0,1}$ $a\_A_{1,2}$ $a\_A_{K,K+1}$
$C(A_0)$ $C(A_1)$ $C(A_2)$ $C(A_K)$
ΔA_0 ΔA_1 ΔA_2 ΔA_K

(b) Insertions "after" a phoneme

Figure 9. Extended insertion states of a metamodel.

```
for $k$ = 0 to K+1 do
    $a\_A_{k,K+1} = \frac{C(A_k)-C(A_{k+1})}{C(A_k)} = \frac{\Delta A_k}{C(A_k)}$
end for
for $k$ = 0 to K do
    $a\_A_{k,k+1} = \frac{C(A_{k+1})}{C(A_k)}$
end for
```

3.3. Weighted finite state transducers

As presented by [25] and [26] , a network of Weighted Finite-State Transducers (WFSTs) is an alternative for the task of estimating W from $\tilde{P}^*$. This is because the speech recognition process can be realized as a composition of WFSTs.

WFSTs can be regarded as a network of automata, each of which accepts an input symbol and outputs one of a finite set of outputs, each of which has an associated probability. The outputs are drawn (in this case) from the same alphabet as the input symbols and can be single symbols, sequences of symbols, or the deletion symbol ϵ . The automata are linked by a (typically sparse) set of arcs and there is a probability associated with each arc.

The WFSTs can be used to model a speaker's phonetic confusions. In addition, a composition of such transducers can model the mapping from phonemes to words, and the mapping from words to a word sequence described by a grammar. The usage proposed here complements and extends the work presented by [20], in which WFSTs were used to correct phoneme recognition errors. Here, the technique is extended to convert noisy phoneme strings into word sequences.

Hence, for this approach, the following transducers to decode $\tilde{P}^*$ into a sequence of words W^* are defined:

1. C, the confusion matrix transducer, which models the probabilities of phoneme insertions, deletions and substitutions.
2. D, the dictionary transducer, which maps sequences of decoded phonemes from $\tilde{P}^* \circ C$ into words in the dictionary.
3. G, the language model transducer, which defines valid sequences of words from D.

Thus, the process of estimating the most probable sequence of words W^* given $\tilde{P}^*$ can be expressed as:

$$W^* = \tau^*(\tilde{P}^* \circ C \circ D \circ G) \tag{6}$$

where τ^* denotes the operation of finding the most likely path through a transducer and $\circ$ denotes composition of transducers [25]. Details of each transducer are presented in the following section.

3.3.1. Confusion matrix transducer C

In this section, the formation of the confusion-matrix transducer C is described. In Section 3.1, $\tilde{p}_j^*$ as the j'th phoneme in $\tilde{P}^*$ and p_j as the j'th phoneme in P were defined, where $Pr(\tilde{p}_j^*|p_j)$

is estimated from the speaker's confusion-matrix, which is obtained from an alignment of many sequences of $\tilde{P}^*$ and P . While single substitutions are modelled in the same way by both metamodels and WFSTs, insertions and deletions are modelled in a different way, taking advantage of the characteristics of the WFSTs. Here, the confusion-matrix transducer C can map single and multiple phoneme insertions and deletions.

P:	/ax	b	aa	th	ih			ax	z	w	ey		ih	ng	dh	ax	b	eh t/
$\tilde{P}^*$:	/ax			r	ih	ng	dh	ax	ng	dh	ax	l	ih	ng	dh	ax	b/	

Table 1. Example of an alignment of transcription P, and recognized output $\tilde{P}^*$, for estimation of a Confusion-Matrix Transducer C.

Consider Table 1, where the top row of phonemes represents the transcription of a word sequence, and the bottom row the output from the speech recognizer. It can be seen that the phoneme sequence */b aa/* is deleted after */ax/*, and this can be represented in the transducer as a multiple substitution/insertion: */ax/*$\rightarrow$*/ax b aa/*. Similarly the insertion of */ng dh/* after */ih/* is modelled as */ih ng dh/*$\rightarrow$*/ih/*. The probabilities of these multiple substitutions / insertions / deletions are estimated by counting. In cases where a multiple insertion or deletion is made of the form $A\rightarrow$*/B C/*, the appropriate fraction of the unigram probability mass *Pr(A*$\rightarrow$*B)* is subtracted and given to the probability *Pr(A*$\rightarrow$*/B C/)*, and the same process is used for insertions or deletions of higher order.

A fragment of the confusion-matrix transducer that represents the alignment of Table 1 is presented in Figure 10. For computational convenience, the weight for each confusion in the transducer is represented as $-logPr(\tilde{p}^*_j|p_j)$. In practice, an initial set of transducers are built directly from the speaker's "unigram" confusion matrix, which is estimated using each transcription/output alignment pair available from that speaker, and then to add extra transducers that represent multiple substitution/insertion/deletions. The complete set of transducers are then determinized and minimized, as described in [25]. The result of these operations is a single transducer for the speaker as shown in Figure 10.

3.3.2. Dictionary and language model transducer (D, G)

The transducer D maps sequences of phonemes into valid words. Although other work has investigated the possibility of using WFSTs to model pronunciation in this component [12], here the pronunciation modelling is done by the transducer C. A small fragment of the dictionary entries is shown in Figure 11, where each sequence of phonemes that forms a word is listed as an FST. The minimized union of all these word entries is also shown. The single and multiple pronunciations of each word were taken from the British English BEEP pronouncing dictionary [13].

The language model FSA (Finite State Automata) G can be represented as in Figure 12, where $< s >$ represents the state for a starting word, and $\{w_1, w_2\}$ a word bigram. A backoff state is added for when a bigram probability is not in the model, in which case $P(w_2|w_1) = P(w_2)B(w_1)$.

In the following sections, applications and results of the techniques reviewed in this section will be presented and discussed.

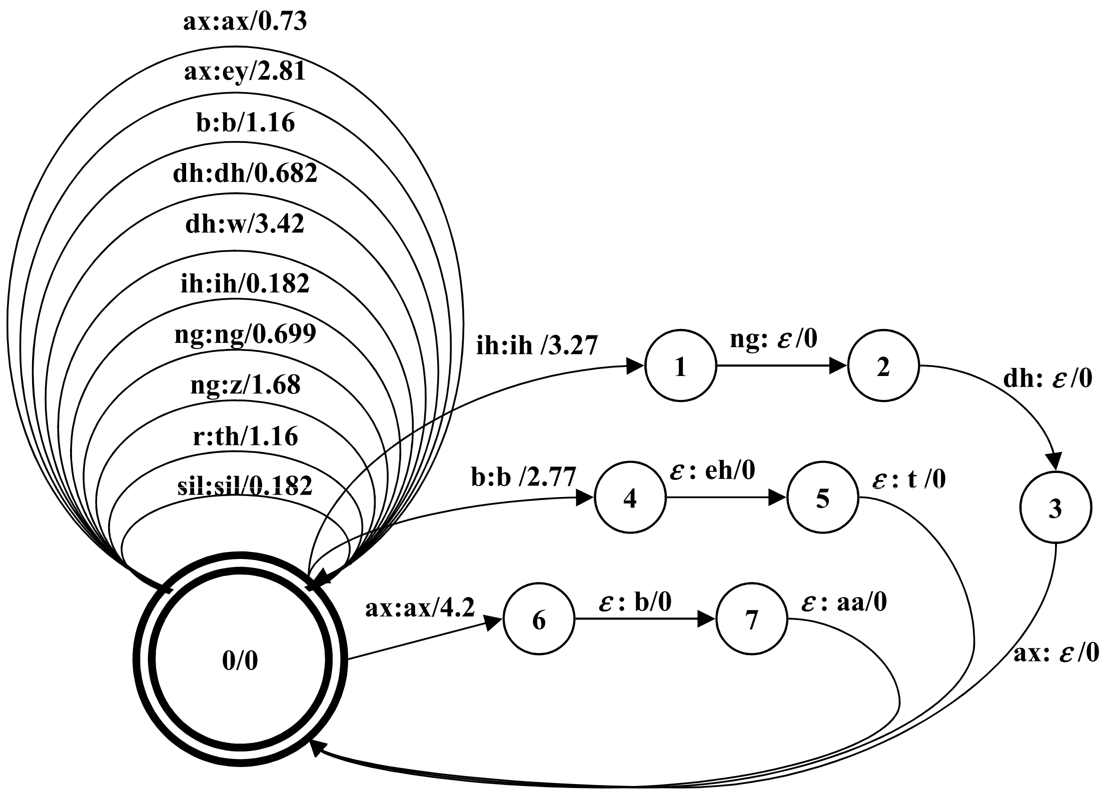


Figure 10. Example of the Confusion-Matrix Transducer *C* for the alignment of Table 1.

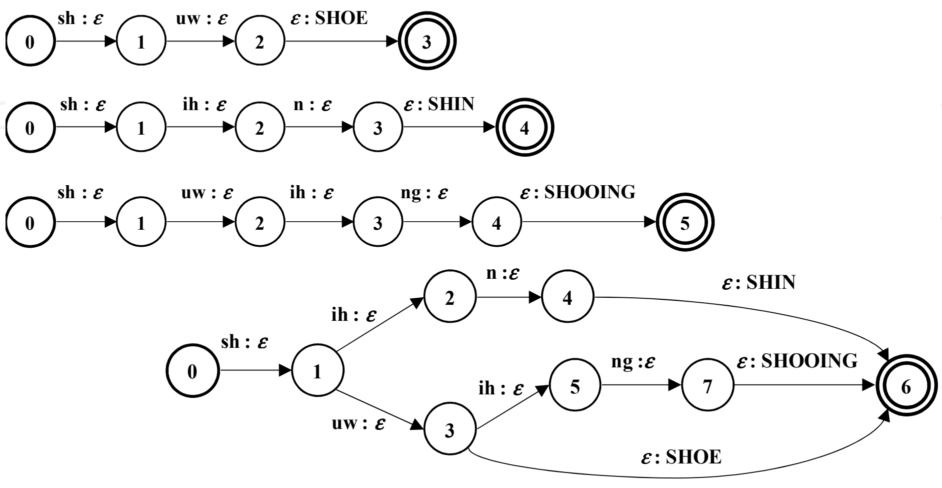


Figure 11. Dictionary transducer *D*: individual entries, and minimized network of entries.

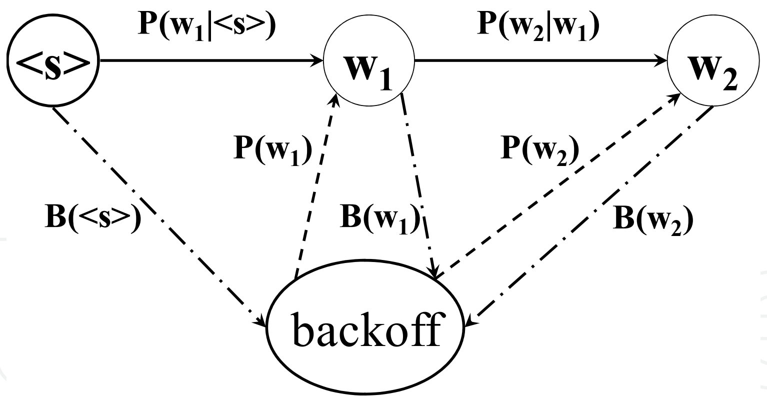


Figure 12. Weighted FSA for a bigram [19].

4. Case studies

4.1. Structure optimization of metamodels with GA

In this section the use of a Genetic Algorithm (GA) to optimize the structure of a set of metamodels and further improve ASR performance is presented [29]. The experiments were performed with speakers from the british-english Wall Street Journal (WSJ) speech database [14], and the metamodels were built with the libraries of the HTK Toolkit[10] from the University of Cambridge.

4.1.1. Chromosome representation

Considering the extended architecture of a metamodel of Figure 8, the length of the insertion contexts and the non-normalized phoneme confusion-matrix are the parameters that will be optimized by the GA. The phoneme confusion-matrix has dimension 47x47 because the database used for this work, the WSCAM0 speech database, consists phonetically of 46 phonemes [13, 14], plus the phoneme $/DEL/$ to identify deletions. The chromosome representation of the parameters of the metamodels is shown in Figure 13.

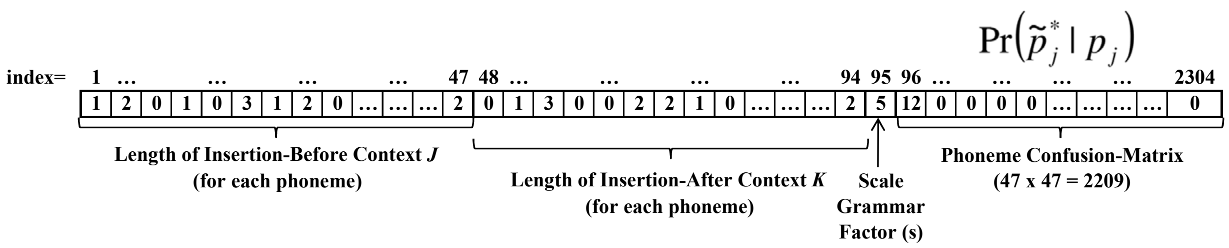

$\Pr(\tilde{p}_j^* \mid p_j)$
index= 1 ... ... 47 48 ... ... 94 95 96 2304
1 2 0 1 0 3 1 2 0 ... ... 2 0 1 3 0 0 2 2 1 0 ... ... 2 5 12 0 0 0 0 0
Length of Insertion-Before Context J (for each phoneme)
Length of Insertion-After Context K (for each phoneme)
Scale Grammar Factor (s)
Phoneme Confusion-Matrix (47 x 47 = 2209)

Figure 13. Chromosome representation of the parameters of the extended metamodels.

Because an insertion context exists for each phoneme, there are 2x47 = 94 insertion contexts in total. This information is arranged in a single vector of dimension 1x94, which is saved in genes 1-94 of the chromosome vector (see Figure 13). In gene 95 the value of the scale grammar factor (s), which controls the influence of the language model over the recognition process, is placed [10]. From genes 96 to 2304 the elements of the confusion-matrix are placed. Hence, each chromosome vector represents all the parameters of the metamodels for all phonemes in the speech set.

The integer values for each set of genes are: K, J = 0 to 3; s = 0 to 10; phoneme confusion - matrix = 0 to 100 (number of occurrences of each aligned pair $\{p_j^*, p_j)\}$ before being normalized as probabilities $Pr(\tilde{p}_j^*|p_j)$).

The general structure of the GA is the one presented in Figure 2. For this, the initial population consists of 10 individuals, where the first element is the initial extended metamodel and the remaining elements are randomly generated within the range of values specified above. Fitness of each individual was measured on the word recognition accuracy (WAcc, Eq. 1) achieved with the resulting metamodels on a training set.

4.1.2. Operators

- **Selection**: The selection method (e.g., how to choose the eligible parents for reproduction) was based on the Roulette Wheel and was implemented as follows:
  - For each of the 10 best individuals in the population, compute its fitness value.
  - Compute the selection probability for each x_i individual as: $p_i = \frac{f_i}{\sum_{k=1}^{N} f_k}$, where N is the size of the population (sub-set of 10 individuals), and f_i the fitness value of the individual x_i.
  - Compute the accumulated probability q_i for each individual as: $q_i = \sum_{j=1}^{i} p_j$.
  - Generate a uniform random number $r \in \{0, 1\}$.
  - If $r < q_i$, then select the first individual (x_1), otherwise, select x_i such that $q_{i-1} < r \leq q_i$.
  - Repeat Steps 4 and 5 N times (until all N individuals are selected).
- **Crossover**: Uniform crossover was used for reproduction of parents chosen by the Roulette Wheel method. A template vector of dimension 1x2304 was used for this, where each of its elements received a random binary value (0, 1). Offspring 1 is produced by copying the corresponding genes from Parent 1 where the template vector has a value of 0, and copying the genes from Parent 2 where the template vector has a value of 1. Offspring 2 is obtained by doing the inverse procedure. 10 offsprings are obtained by crossover from the 10 individuals in the initial population. This increases the size of the population to 20.
- **Mutation**: The mutation scheme consisted in randomly changing values in the best 5 individuals as follows:
  - change 5 genes in the sub-vector that includes the length of the insertion contexts (genes 1 to 94);
  - change the scale grammar factor (gene 95);
  - change 94 genes in the sub-vector that represents the phoneme confusion-matrix (genes 96 to 2304).

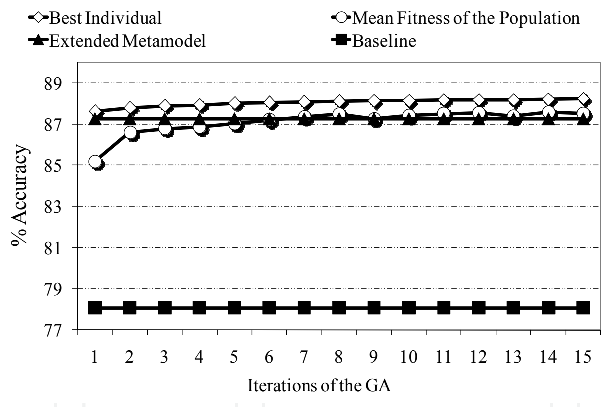
  In this way, a population of 25 individuals is obtained for the GA.
- **Stop condition**: The GA is repeated until convergence through generations is achieved. A boundary of 15 iterations was set for the GA as it was observed that minimum variations were observed after that.

4.1.3. Experiments and results

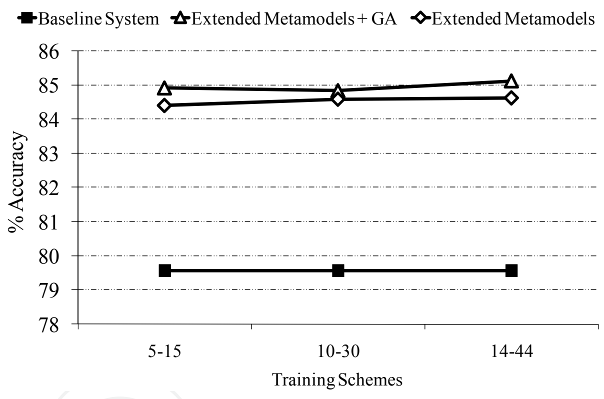
A speaker-independent ASR system was built with the HTK Toolkit [10] using the WSJCAM0 speech database [14]. 92 speakers from the set *si_tr* were used for acoustic model training (three-state left-to-right HMMs, with eight Gaussian components per state). The front-end used 12 MFCCs plus energy, delta, and acceleration coefficients.

The experiments were done with speech data from 10 speakers of the development set *si_dt* of the same database. From each speaker, 74 sentences were selected, discarding adaptation sentences that were the same for all speakers. From this set, 30 were selected for testing-only purposes and 44 for confusion-matrix estimation (training of metamodels) and fitness evaluation for the GA. The metamodels were trained with three different sets of sentences: 5, 10, and 14 sentences, and fitness evaluation and metamodels re-estimation were performed with 15, 30, and 44 sentences respectively. This formed the *training schemes 5-15*, *10-30*, and *14-44* (e.g., if 5 sentences were used for metamodel training, fitness evaluation of the GA and metamodel re-estimation were performed with 15 sentences). Word bigram language models were estimated from the transcriptions of the *si_dt* speakers, and were the same for all the systems (baseline and metamodels).

Figure 14(a) shows the graph of fitness convergence of the GA when the scheme *14-44* was used. The baseline performance is around 78% while the initial extended metamodel achieved a performance of 87.3%. The mean fitness of the populations starts at 85% and, as the individuals evolve, increases to up to 87.4%. The best individual from each generation achieves a fitness of up to 88.20%. For the metamodels trained with the schemes *5-15* and *10-30* the pattern of converge was very similar.



(a) Convergence of the GA.



(b) Performance on test data.

Figure 14. Results of the GA optimized metamodels.

Figure 14(b) shows the mean performance of the optimized metamodels on the testing set when all training schemes were used. On average, the optimized metamodels showed an increase of around 0.50% when compared with the initial extended metamodels. These gains were statistically significant as shown in Table 2. The matched pairs test described by [32] was used to test for statistical significant difference.

When the extended metamodels were optimized with a GA, a significant gain was achieved with schemes *5-15* and *14-44*. The use of the GA also reduced the size of the metamodels by 10% for all cases, thus eliminating unnecessary states and transitions. Finally, with a used vocabulary of 3000 words and 740 sentences (300 for testing), there is an important confidence about the significance of these results.

Training Scheme	Metamodel	Errors in Test Set	p -value	Conclusion
5-15	Initial	773	0.0591	<0.1 Significant
	Optimized	748		
10-30	Initial	764	0.2479	Not Significant
	Optimized	752		
14-44	Initial	761	0.0567	<0.1 Significant
	Optimized	737		

Table 2. Significance test for the initial and the optimized extended metamodels.

4.2. Estimation of phoneme confusion patterns with NMF

In this section, an application of the NMF approach to phoneme confusion-matrix estimation to improve ASR performance is presented [30]. The NMF estimates are integrated into the original metamodels (see Figure 7), and in this application, the NMF technique is extended to estimate also insertion patterns. The results obtained with NMF show statistically significant gains in recognition accuracy when compared with an adapted baseline ASR system and the performance of the original metamodels.

The procedure presented in Figure 6 was used for confusion-matrix estimation. A threshold $CT = 2$ was set for smoothing purposes, and the partial confusion-matrices CM_U^y were estimated from accurate alignments of P and $\tilde{P}^*$ from training speech data.

4.2.1. Experiments and results

The Wall Street Journal (WSJ) database, and the baseline ASR system described in Section 4.1.3 were used for the NMF experiments. The training-speakers S_x for V (see Section 3.1.1) consisted of 85 speakers from the *si_tr* set, which were also used to estimate $\overline{CM}$. 10 test-speakers S_y for V were selected from the set *si_dt* of the same database. Note that from the training-speakers, only **target** confusion-matrices CM^x were estimated. For the test-speakers, **partial** CM_U^y and **target** CM^y confusion-matrices were estimated in order to evaluate the quality of the NMF estimates of $\widehat{CM}^y$.

Supervised Maximum Likelihood Linear Regression (MLLR) adaptation [33] was implemented using the same sets of utterances U selected for confusion-matrix estimation. A regression class tree with 32 terminal nodes was used for this purpose. As shown in Table 3, the mean number of MLLR transformations increased as more U utterances were used. The adapted acoustic models represent the baseline for the experiments.

Adaptation Data (U)	5	10	15	20	30
Mean No. of MLLR Transformations	4	7	10	11	12

Table 3. Mean number of MLLR transformations across all test speakers using different sets of adaptation data.

A word-bigram language model, estimated from the data of the *si_tr* speakers, was used to obtain $\tilde{P}^*$ and estimate CM^x with the unadapted baseline system. In order to keep these sequences independent from those of the test-set speakers, the word-bigram language model used to decode $\tilde{P}^*$ for CM^y and CM_U^y, was estimated from the data of the selected test-speakers of the *si_dt* set. In all cases, a grammar scale factor s of 10 was used. The metamodels were tested using all the utterances available from the speakers S_y.

Figure 15 shows the performance of the metamodels using MLLR adapted acoustic models. The metamodels trained with **partial** estimates improved over the adapted baseline. When the NMF estimates were used, the accuracy of the metamodels improved when the training data was small. As presented in Table 4, these improvements were statistically significant when 5, 10, and 15 utterances were used for training/estimation.

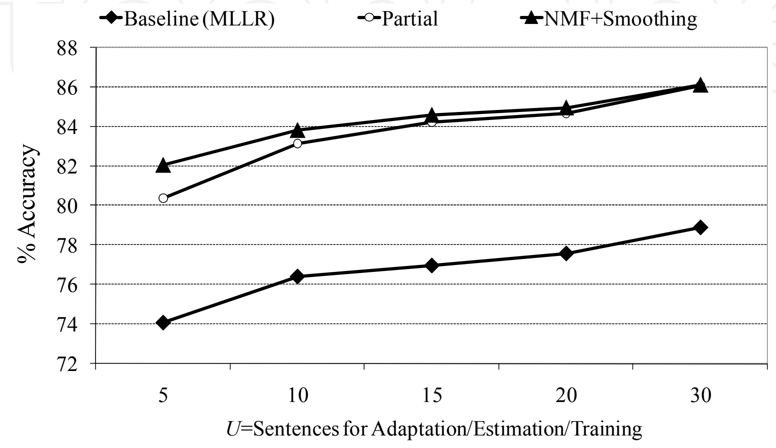


Figure 15. Mean word recognition accuracy of the metamodels and the adapted baseline across all test-speakers.

U	Metamodel	Errors in Test Set	p-value	Conclusion
5	Partial	2608	0.000000	< 0.001, 0.01, 0.05, 0.1 Significant
	NMF	2383		
10	Partial	2240	0.005603	< 0.01, 0.05, 0.1 Significant
	NMF	2151		
15	Partial	2093	0.102104	< 0.105 Significant
	NMF	2045		
20	Partial	2033	0.198040	Not Significant
	NMF	1996		
30	Partial	1847	0.888571	Not Significant
	NMF	1843		

Table 4. Significance test for the metamodels trained with partial and NMF estimates.

4.3. Metamodels and WFSTs to improve ASR performance for disordered speech

Dysarthria is a motor speech disorder characterized by weakness, paralysis, or poor coordination of the muscles responsible for speech. Although ASR systems have been developed for disordered speech, factors such as low intelligibility and limited phonemic repertoire decrease speech recognition accuracy, making conventional speaker adaptation algorithms perform poorly on dysarthric speakers.

In the work presented by [22], rather than adapting the acoustic models, the errors made by the speaker are modelled to attempt to correct them. For this, the metamodels and WFSTs techniques are applied to correct the errors made at the phonetic level and make use of a language model to find the best estimate of the correct word sequence. Experiments performed with dysarthric speech of different levels of severity showed that both techniques outperformed standard adaptation techniques.

4.3.1. Limited phonemic repertoire

Among the identified factors that give rise to ASR errors in dysarthric speech [16], the most important are decreased intelligibility (because of substitutions, deletions and insertions of phonemes), and limited phonemic repertoire, the latter leading to phoneme substitutions. To illustrate the effect of reduced phonemic repertoire, Figure 16 shows an example phoneme confusion-matrix for a dysarthric speaker from the NEMOURS Database of Dysarthric Speech (described in Section 4.3.2). This confusion-matrix is estimated by a speaker-independent ASR system, and so it may show confusions that would not actually be made by humans. From Figure 16 the following observations are made:

- A small set of phonemes (in this case the phonemes */ua/*, */uw/*, */m/*, */n/*, */ng/*, */r/*, and */sil/*) dominates the speaker's output speech.
- Some vowel sounds and the consonants */g/*, */zh/*, and */y/*, are never recognized correctly. This suggests that there are some phonemes that the speaker apparently cannot enunciate at all, and for which he or she substitutes a different phoneme, often one of the dominant phonemes mentioned above.

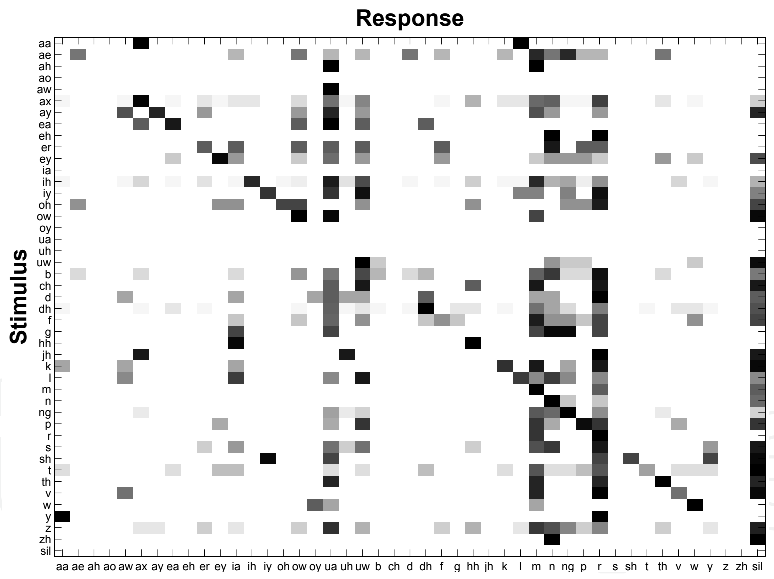


Figure 16. Phoneme confusion-matrix from a dysarthric speaker.

These observations differ from the pattern of confusions seen in a normal speaker as shown in Figure 4. This confusion-matrix shows a clearer pattern of correct recognitions, and few confusions of vowels with consonants.

Most speaker adaptation algorithms are based on the principle that it is possible to apply a set of transformations to the parameters of a set of acoustic models of an "average" voice to move them closer to the voice of an individual (e.g., MLLR). Whilst this has been shown to be successful for normal speakers, it may be less successful in cases where the phoneme uttered is not the one that was intended but is substituted by a different phoneme or phonemes, as often happens in dysarthric speech. In this situation, it is suggested that a more effective approach is to combine a model of the substitutions likely to have been made by the speaker with a language model to infer what was said. So rather than attempting to adapt the system, we model the insertion, deletion, and substitution errors made by a speaker and attempt to correct them.

4.3.2. Speech data, baseline recognizer, and adaptation technique

While the baseline ASR system was built with the WSJ database as in [29] and [30], the dysarthric speech data was provided by the NEMOURS Database [21]. This database is a collection of 814 short sentences spoken by 11 speakers (74 sentences per speaker) with varying degrees of dysarthria (data from only 10 speakers was used as some data is missing for one speaker). The sentences are nonsense phrases that have a simple syntax of the form "the X is Y the Z", where X and Z are usually nouns and Y is a verb in present participle form (for instance, the phrases "The shin is going the who", "The inn is heaping the shin", etc.). Note that although each of the 740 sentences is different, the vocabulary of 112 words is shared.

Speech recognition experiments were implemented by using the baseline recognizer on the dysarthric speech. For these experiments, a word-bigram language model was estimated from the (pooled) 74 sentences provided by each speaker.

The technique used for the speaker adaptation experiments was MLLR (Maximum Likelihood Linear Regression) [33]. From the complete set of 74 sentences per speaker, 34 sentences were used for adaptation and the remaining 40 for testing. The set of 34 was divided into sets to measure the performance of the adapted baseline system when using a different amount of adaptation data. Thus adaptation was implemented using 4, 10, 16, 22, 28, and 34 sentences. For future reference, the baseline system adapted with X sentences is termed as MLLR_X, and the baseline without any adaptation as BASE.

An experiment was done to compare the performance of the baseline and MLLR-adapted recognizer (using 16 utterances for adaptation) with a human assessment of the dysarthric speakers used in this study. Recognition was performed with a grammar scale factor and word insertion penalty as described in [10].

Figure 17 shows the intelligibility of each of the dysarthric speakers as measured using the Frenchay Dysarthria Assessment (FDA) test in [21], and the recognition performance (% Accuracy) when tested on the unadapted baseline system (BASE) and the adapted models (MLLR_16). The correlation between the FDA performance and the recognizer performance is 0.67 (unadapted models) and 0.82 (adapted). Both are significant at the 1% level, which gives some confidence that the recognizer displays a similar performance trend when exposed to different degrees of dysarthric speech as humans.

4.3.3. Experiments and results

The metamodels used in this case had the original architecture which was shown in Figure 7. In Figure 18 the results of the metamodels on the phoneme strings from the MLLR adapted acoustic models are shown. When a very small set of sentences, e.g., 4, is used for training of the metamodels, it is possible to get an improvement of approximately 1.5% over the MLLR

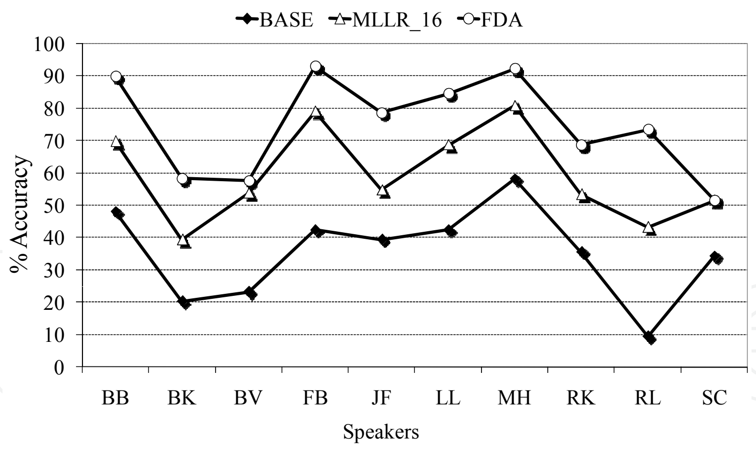


Figure 17. Comparison of recognition performance: Human assessment (FDA), unadapted (BASE) and adapted (MLLR_16) SI models.

adapted models. This gain in accuracy increases as the training/adaptation data is increased, obtaining an improvement of almost 3% when all 34 sentences are used.

The matched pairs test described by [32] was used to test for significant differences between the recognition accuracy using metamodels and the accuracy obtained with MLLR adaptation when a certain number of sentences were available for metamodel training. The results with the associated p-values are presented in Table 5.

In all the cases, metamodels improve MLLR adaptation with p-values less than 0.01 and 0.05. Note that the metamodels trained with only four sentences (META_04) decrease the number of word errors from 1174 (MLLR_04) to 1139.

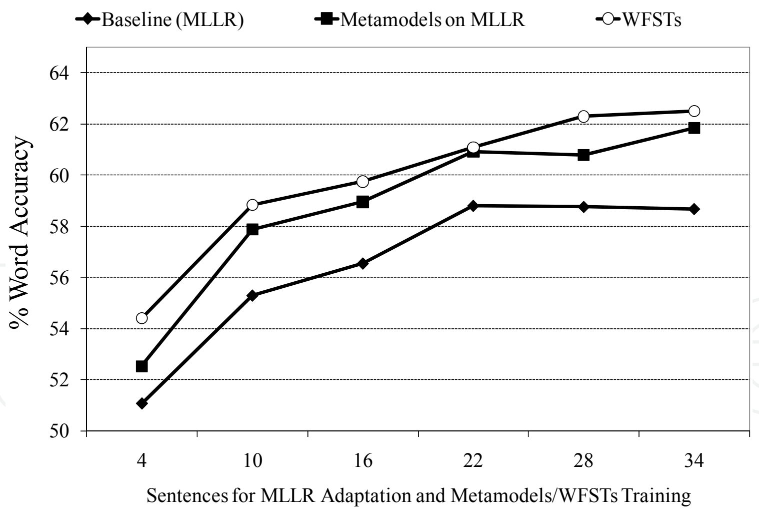


Figure 18. Mean word recognition accuracy of the MLLR adapted baseline, the metamodels, and the WFSTs across all dysarthric speakers.

The WFSTs were tested using the same conditions and speech data as the metamodels. The FSM Library [25] from AT&T was used for the experiments with WFSTs. Figure 18 shows clearly the gain in performance given by the WFSTs over both MLLR and the metamodels. This gain is statistically significant at the 0.1 level for all cases except when 22 and 34 sentences were used for training.

System	Errors in Test Set	p-value	Conclusion
MLLR_04	1174	0.00168988	< 0.01 Significant
META_04	1139		
MLLR_10	1073	0.00024590	< 0.001 Significant
META_10	1036		
MLLR_16	1043	0.00204858	< 0.01 Significant
META_16	999		
MLLR_22	989	0.00003510	< 0.001 Significant
META_22	941		
MLLR_28	990	0.00240678	< 0.01 Significant
META_28	952		
MLLR_34	992	0.00000014	< 0.001 Significant
META_34	924		

Table 5. Comparison of statistical significance of results over all dysarthric speakers using metamodels.

5. Conclusions

In this chapter the details and applications of techniques to improve ASR performance were presented. These consisted in heuristic and statistical post-processing techniques which made use of a speaker's phoneme confusion - matrix for error modelling, achieving correction at the phonetic level of an ASR's output for both, normal and disordered speech.

The first post-processing technique, termed as metamodels, incorporated the information of the speaker's confusion-matrix into the recognition process. The metamodels expanded the confusion-matrix modelling by incorporating information of the pattern of insertions associated with each phoneme. Deletions were modelled as a phoneme being substituted (or confused) by the "DELETION" symbol.

A metamodel's architecture, for which there was an extended version, was suitable for further optimization, and in Section 4.1 the application of a GA for this purpose was presented. The improvements in word recognition accuracy obtained on normal speech after optimization were statistically significant when compared with the previous performance of the metamodels. This corroborated the findings of other works [8, 9], where GA was applied to improve recognition performance by evolving the internal parameters of an HMM (state transitions, observation probabilities).

Also, in Section 4.2 was presented another application case, where the use of Non - negative Matrix Factorization (NMF) provided more accurate estimates of a speaker's phoneme confusion - matrix which, when incorporated into a set of metamodels, improved significantly the accuracy of an ASR system. This performance was also higher than the performance of the metamodels trained with original partial data (with no NMF estimates).

However, when tested with speakers with disordered speech, where significant increase in substitution, deletion, and insertion errors exists, two important issues were identified:

- The metamodels were unable to model specific phoneme sequences that were output in response to individual phoneme inputs. They were capable of outputting sequences, but the symbols (phonemes) in these sequences were conditionally independent, and so specific sequences could not be modelled. This also led to be unable to model deletion of sequences of phonemes, although the original and extended architecture ensures modelling of multiple insertions and single substitutions / deletions.

- Adding the "DELETION" symbol led to an increase in the size of the dictionary, because it could potentially substitute each phoneme in the network of metamodels during the recognition process. Not adding this symbol led to the problem of a "Tee" model in the HTK package [10]. In such case, a deletion is represented as a direct transition from the initial to the final state, thus allowing "skipping" the phoneme. The decoding algorithm failed because it was possible to traverse the complete network of metamodels without absorbing a single input symbol.

Hence, for modelling of deletions, a specific way to define a "missing" or "empty" observation was needed. An alternative to solve these issues, the second post-processing technique, a network of Weighted Finite State Transducers (WFSTs) was presented. In Section 4.3, a case of improving ASR performance for speakers with the speech disorder of dysarthria was presented, where both techniques, metamodels and WFSTs were used for that purpose. The main advantage of the WFSTs is the definition and use of the *epsilon* (ϵ) symbol, which in finite-state automata represents an "empty" observation. This allowed the modelling of multiple deletions, substitutions, and insertions. General improvement on ASR for dysarthric speech was obtained with this technique when compared to the metamodels and the adapted baseline system.

As conclusion it can be mentioned that the techniques presented in this chapter offer wide possibilities of application in the general field of speech recognition. Their robustness has been corroborated with case studies with normal and disordered speech, where higher performance was consistently achieved over unadapted and adapted baseline ASR systems. Also, these techniques are flexible enough to be optimized and integrated with other processes as in [30] and [5].

Author details

Santiago Omar Caballero Morales
Technological University of the Mixteca, Mexico

6. References

[1] Jurafsky, D. & Martin, J.H. (2009). *Speech and Language Processing*, Pearson: Prentice Hall, USA.

[2] Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co., USA.

[3] Chan, C.W.; Kwong, S.; Man, K.F. & Tang, K.S. (2001). Optimization of HMM Topology and its Model Parameters by Genetic Algorithms. *Pattern Recognition*, Vol. 34, pp. 509-522

[4] Hong, Q. Y. & Kwong, S. (2005). A Genetic Classification Method for Speaker Recognition. *Engineering Applications of Artificial Intelligence*, Vol. 18, pp. 13-19

[5] Matsumasa, H.; Takiguchi, T.; Ariki, Y.; LI, I-C. & Nakabayash, T. (2009). Integration of Metamodel and Acoustic Model for Dysarthric Speech Recognition. *Journal of Multimedia - JMM*, Vol. 4, No. 4, pp. 254-261

[6] Lee, D.D. & Seung, H.S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, Vol. 401, pp. 788-791

[7] Lee, D.D. & Seung, H.S. (2001). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, Vol. 13, pp. 556-562

[8] Takara, T.; Iha, Y. & Nagayama, I. (1998). Selection of the Optimal Structure of the Continuous HMM using the Genetic Algorithm, *Proc. of the International Conference on Spoken Language Processing (ICSLP 98)*, Sydney, Australia.

[9] Xiao, J.; Zou, L. & Li, C. (2007). Optimization of Hidden Markov Model by a Genetic Algorithm for Web Information Extraction, *Proc. of the International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2007)*, Chengdu, China, ISBN: 978-90-78677-04-8

[10] Young, S. & Woodland, P. (2006). *The HTK Book (for HTK Version 3.4)*, Cambridge University Engineering Department, United Kingdom.

[11] Reeves, C.R. (1993). *Modern Heuristic Techniques for Combinational Problems*, John Wiley & Sons Inc., USA.

[12] Bodenstab, N. & Fanty, M. (2007). Multi-pass pronunciation adaptation, *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2007*, Honolulu, Hawaii, Vol. 4, pp. 865–868, ISBN: 1-4244-0728-1

[13] Robinson, T.; Mitton, R.; Wilson, M.; Foote, J.; James, D. & Donovan, R. (1997). *British English Example Pronunciation Dictionary (BEEP) v1.0*, University of Cambridge, Department of Engineering, Machine Intelligence Laboratory, United Kingdom

[14] Robinson, T.; Fransen, J.; Pye, D.; Foote, J. & Renals, S. (1995). WSJCAM0: A British English speech corpus for large vocabulary continuous speech recognition, *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1995*, Detroit, USA, Vol.1., pp. 81-84, ISBN: 0-7803-2431-5

[15] Hamilton, H. J. (2007). *Notes of Computer Science 831: Knowledge Discovery in Databases*, University of Regina, Department of Computing Sciences, Canada.

[16] Rosen, K. & Yampolsky, S. (2000). Automatic speech recognition and a review of its functioning with dysarthric speech. *Augmentative and Alternative Communication*, Vol. 16, pp. 48-60

[17] Green, P.; Carmichael, J.; Hatzis, A.; Enderby, P.; Hawley, M.S. & Parker M.(2003). Automatic Speech Recognition with Sparse Training Data for Dysarthric Speakers, *Proc. of the 8th European Conference on Speech Communication Technology (Eurospeech)*, Geneva, Switzerland, pp. 1189-1192, ISSN: 1018-4074

[18] Torre, D.; Villarrubia, L.; Hernandez,L. & Elvira, J.M. (1997). Automatic Alternative Transcription Generation and Vocabulary Selection for Flexible Word Recognizers, *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2007*, Honolulu, Hawaii, Vol. 2, pp. 1463-1466

[19] Fosler-Lussier, E. (2008). *Finite State Machines In Spoken Language Processing (FSM Tutorial)*, Speech and Language Technology Laboratory, Ohio State University, USA.

[20] Levit, M.; Alshawi, H.; Gorin, A. & Nöth, E. (2003). Context-Sensitive Evaluation and Correction of Phone Recognition Output, *Proc. of the 8th European Conference on Speech Communication Technology (Eurospeech)*, Geneva, Switzerland, pp. 925-928, ISSN: 1018-4074

[21] Bunnel, H.T.; Polikoff, J.B.; Menéndez-Pidal, X.; Peters, S.M. & Leonzio, J.E. (1996). The Nemours Database of Dysarthric Speech, *Proc. of the Fourth International Conference on Spoken Language Processing (ICSLP 96)*, Philadelphia, USA.

[22] Caballero-Morales, S.O. & Cox, S.J. (2009). Modelling Errors in Automatic Speech Recognition for Dysarthric Speakers. *EURASIP Journal on Advances in Signal Processing*, pp. 1-14, ISSN: 1687-6172

[23] Li, Y.X.; Tan, C.L.; Ding, X. & Liu, C. (2004). Contextual post-processing based on the confusion matrix in offline handwritten Chinese script recognition. *Pattern Recognition*, Vol. 37, No. 9, pp. 1901-1912

[24] Cox, S.J. & Dasmahapatra, S. (2002). High level approaches to confidence estimation in speech recognition. *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 7, pp. 460-471,. ISSN: 1063-6676

[25] Mohri, M.; Pereira, F. & Riley, L. (2002). Weighted finite state transducers in speech recognition. *Computer Speech and Language*, Vol. 16, pp. 69-88, ISSN: 0885-2308

[26] Fosler-Lussier, E.; Amdal, I. & Kuo, H.-K.J. (2002). On the road to improved lexical confusability metrics, *ISCA Tutorial and Research Workshop on Pronunciation Modelling and Lexicon Adaptation (PMLA-2002)*, Estes Park, Colorado, USA.

[27] Thatphithakkul, N. & Kanokphara, S. (2004). HMM Parameter Optimization using Tabu Search, *International Symposium on Communications and Information Technologies (ISCIT) 2004*, Sapporo, Japan, pp. 904-908

[28] Cox, S. J. (2008). On Estimation of A Speaker's Confusion Matrix from Sparse Data, *Proc. of the 9th Annual Conference of the International Speech Communication Association (Interspeech 2008)*, Brisbane, Australia, pp. 2618-2621, ISSN: 1990-9772

[29] Caballero-Morales, S.O. (2011). Structure Optimization of Metamodels to Improve Speech Recognition Accuracy, *Proc. of the International Conference on Electronics Communications and Computers (CONIELECOMP) 2011*, pp. 125-130, ISBN: 978-1-4244-9557-3

[30] Caballero-Morales, S.O. & Cox, S.J. (2009). On the Estimation and the Use of Confusion-Matrices for Improving ASR Accuracy, *Proc. of the International Conference on Spoken Language Processing (Interspeech 2009)*, pp. 1599-1602, ISSN: 1990-9772

[31] Caballero-Morales, S.O. & Cox, S.J. (2007). Modelling confusion matrices to improve speech recognition accuracy, with an application to dysarthric speech, *Proc. of the International Conference on Spoken Language Processing (Interspeech 2007)*, pp. 1565-1568, ISSN: 1990-9772

[32] Gillick, L. & Cox, S.J. (1989). Some statistical issues in the comparison of speech recognition algorithms, *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1989*, Glasgow , United Kingdom, pp. 532-535, ISSN: 1520-6149

[33] Leggetter, C.J. & Woodland, P.C. (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, Vol. 9, No. 2, pp. 171-185, ISSN: 0885-2308