Word Error Rate Improvement and Complexity Reduction in Automatic Speech Recognition by Analyzing Acoustic Model Uncertainty and Confusion

Andi Buzo, Horia Cucu, Corneliu Burileanu
Faculty ETTI
University "Politehnica" of Bucharest
Bucharest, Romania
cburileanu@messnet.pub.ro

Miruna Paşca[1], Vladimir Popescu[2]
[1]Faculty ETTI
University "Politehnica" of Bucharest
Bucharest, Romania
[2]IRIT Laboratory, "Paul Sabatier" University
Toulouse, France

Abstract—**In this paper, a study about the uncertainty of the trained acoustic models and the confusion among these models is made in the context of speech recognition. The purpose is to find the most relevant voice features, hence the analysis is made on a per-feature basis. *Model uncertainty* is defined as a measure of feature distribution overlapping. A model is compared only to the models it is more similar to. Hence, confusion matrices are built from both feature distributions and recognition results. Next, the voice features are weighted according to their relevance in order to increase the discrimination among models, while relevance itself is deduced from the values of model uncertainty. Experimental results show that, by appropriate weighting, the recognition accuracy, in terms of Word Error Rate (WER), improves. Moreover, by removing the features with lower weights, the recognition accuracy is maintained, but the number of calculations is significantly reduced.**

Keywords—Automatic Speech Recognition; Acoustic Model Uncertainty; Model Confusion

I. INTRODUCTION

Many recent studies in Automatic Speech Recognition (ASR) are focused on both reducing the processing power needed by the recognition algorithm and increasing the recognition accuracy. Moreover, the goal to implement ASR systems on portable devices has stimulated studies in reducing ASR system complexity under real-world conditions because of limited processing power and noisy environments they are used in, or because of the low-quality microphones they are equipped with. Many approaches on reducing system complexity consist in quantization techniques [4], Hidden Markov Model (HMM) parameter tying [5] and developing fixed-point algorithms, as in [6], [7]. As a result of such studies, robust systems for limited processing power devices have been developed, such as the PocketSphinx [8], PocketSUMMIT [9] and IBM systems [10].

In this paper, we are proposing a novel approach which reduces the number of calculations in the ASR process and also improves the recognition rate. The approach is based on the analysis of the phoneme overlap in the feature space. The spectra overlap of two phonemes is calculated from the trained Hidden Markov Models (HMMs). The method starts from the assumption that the features do not have the same relevance in discriminating the phonemes. For each phoneme, a per-feature analysis is made in order to see how much the feature distribution overlaps with other phoneme distributions. It is considered that the features which overlap less are more relevant and better characterize the phoneme. Eventually, features are given weights according to their relevance. Experimental results show that applying weights will improve the recognition accuracy. They also show that removing the least relevant features for a phoneme will maintain the recognition accuracy. Feature removal leads to the reduction of the number of calculations in the decoding process.

The method of weighting the features according to their relevance is used in many other studies, like in data missing algorithms and in uncertainty decoding [1], [11], [12]. However, in these algorithms the relevance of the features is measured after the feature enhancement or noise removal process. The features are considered reliable or corrupted. In our approach, the relevance is measured based on the inherent overlap that exists among the spectra of different phonemes. In similar studies like in [2] and [3], uncertainty among models is analyzed and Support Vector Machines are used to define hyper-planes that separate model classes.

The main issue when calculating weights for the features is that a feature can help distinguish one phoneme from one group of phonemes, but it can be useless in distinguishing it from another group of phonemes. For this purpose, a confusion matrix is built. It helps to find what phonemes are confused the most. Hence, for each phoneme, the feature weights are calculated based on the spectra overlap with the phonemes it is more often confused with. Confusion is used in several works [13], [14], [15], in different contexts, but with the same purpose of identifying the pairs or group of phonemes that are confused with each other.

In a similar study [16], the entropy is used as a measure in order to determine the most relevant features in the recognition process. This technique gives good results for speech signals in noisy environments. However, they involve the calculation of the entropy at any time frame for each feature, and this increases the processing time. Thus, the advantage of our approach is that all the calculations are made offline and only the multiplication with the weights is made during the recognition process. The evaluation of the performance in [16] is made with the AURORA 2 framework [17], which contains noisy speech recordings, whereas in our work we use a database in the Romanian language, hence a direct comparison cannot be made.

Experimental results prove that the recognition accuracy increases in terms of WER by using appropriate weights for voice features. On the other hand, removing the less relevant features, the number of calculations is reduced and the WER is maintained. Experiments are performed with a database in the Romanian language and with an ASR system based on the HTK toolkit [18].

II. PROPOSED APPROACH

A. Definition of model uncertainty

In the classical HMM framework, the decision in the recognition process is made based on the scores obtained by each model for a given set of observations. The scores for each frame are calculated as sums of log-likelihoods given by each of the voice features. However some features can be more relevant than others. Here, relevance is seen as the ability to distinguish among models. Obviously, for two given models, the higher the overlapping of state output distributions of a feature, the lower the ability of this feature to discriminate them. In this work, the overlapping of distributions is referred to as model uncertainty. In Fig. 1, we exemplify the evolution over time, of a distribution associated to a given feature for two models and their overlapping. Section 2.4 describes in detail how model uncertainty is measured.

Ideal trained models do not have overlapping distributions. This is not possible because phonemes themselves have inherently common spectral regions. However, a pair of models overlaps more than another pair and not all the feature distributions overlap in the same way.

B. Types of model uncertainty

We classify model uncertainty in three types according to their causes:

Type 1: When the corresponding feature takes similar values for the two models being compared and, consequently, the two distributions overlap even though they might be well-trained and robust (Fig. 2). Hence, if during decoding the feature takes values from the overlapping region, the models will yield similar scores.

Type 2: When the corresponding feature, for a given model, takes values in the whole range (Fig. 3). This means that this feature does not characterize the respective model, hence the score that it yields in every frame has no relevance.

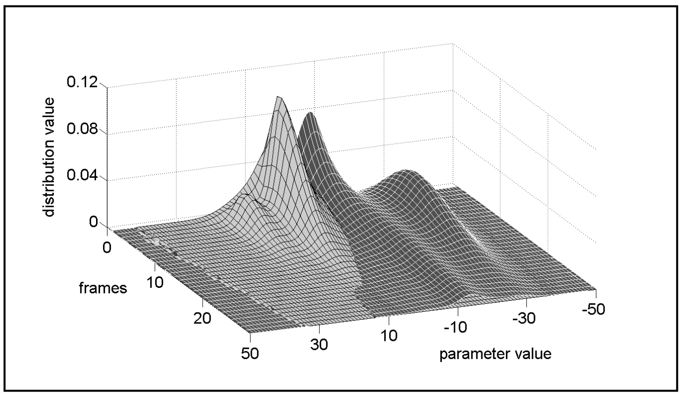

Figure 1. The overlapping of two feature distribution evolutions over time.

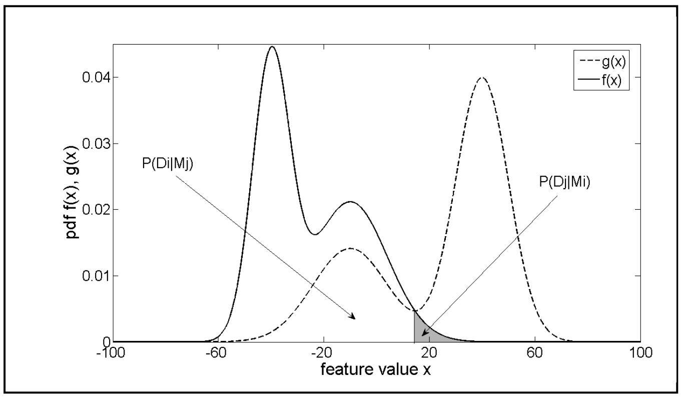

Figure 2. Model uncertainty of type 1 and error probability in making decisions. The grayed area represents *P(Dj|Mi),* the probability of deciding erroneously that model *j* is detected instead of model *i* (section2.4).

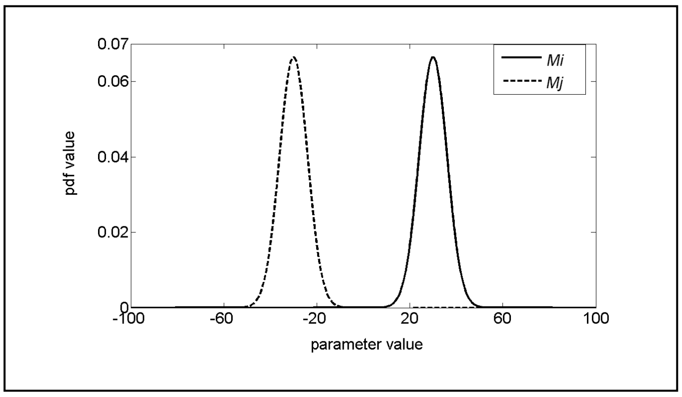

Figure 3. Model uncertainty of type 2

Type 3: When the current observation value does not fall within either of the distribution ranges (Fig. 4). In this case, the distribution closer to the observation value will have a higher score than the other one, thus it is reasonable that both models get the same (low) score.

The above classification assumes that models are well-trained and the observations are not distorted. Any distortion while training or decoding, however, can lead to one of the above situations. For example, if the training database contains

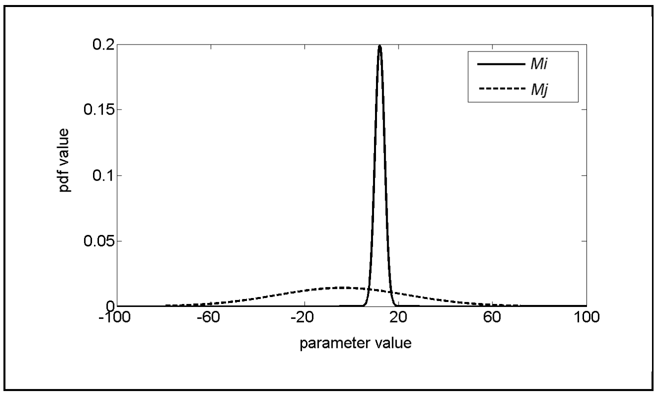

Figure 4. Model uncertainty of type 3

errors, this can cause a feature distribution to cover the whole range of values as in the case of the confusion of the Type 2 (Fig. 3). While decoding, it does not matter if the cause is the erroneous database or the feature just takes values in the whole range; the effect is that this feature will not yield good decisions and, hence, must have a lower weight. An error in the observation sample can lead to Type 1 confusion by placing the feature in the overlapping region (Fig. 2), or it can lead to Type 3 confusion by placing the feature outside the two distributions. In this paper, only model uncertainty of the first type is investigated. In the first attempts on the second type, the results obtained were not promising, because only a few distributions (less than 1%) obtained with our database are spread as in Fig. 3. Hence, by applying weights calculated based on model uncertainty of second type, WER did not decrease. Correcting model uncertainty of the third type does not modify the score significantly either. The error in the score caused by the Type 3 model uncertainty is not sufficient to lead to a wrong decision.

In Fig. 2, Fig. 3 and Fig. 4 the probability density function (*pdf*) values are represented on the ordinate axis and *Mi* and *Mj* are the feature distributions for model *i* and *j* respectively.

C. Building the distribution evolution from the trained HMMs

Distributions are built from the already-trained HMMs. Embedded training is used with the Baum-Welch algorithm [18]. A distribution evolution will be generated by calculating the distribution for each frame. The distribution for one frame (of model *i*) is calculated by summing over the distributions of all states, weighted by the probability that the HMM is in that state in that frame (i.e. $p_s(t)$):

$$d_t^{(i)}(x) = \sum_{s=1}^{S} d_s^{(i)}(x) \cdot p_s(t) \tag{1}$$

where $d_s^{(i)}(x)$ is the *pdf* of the output of state s; S represents the total number of states. In the case of a Gaussian Mixture Model (GMM) as the state's output probability distribution, the *pdf* is calculated with the formula:

$$d_s^{(i)}(x) = \sum_{g=1}^{G} w_{sg} N(x, \mu, \sigma) \tag{2}$$

where $N(x,\mu,\sigma)$ is the normal distribution of mean μ and variance σ^2, and w_{gs} is the weight of mixture g (the number of mixtures is G) of state s. The probability that state s of an HMM emits in frame t is calculated by the recursion:

$$p_s(t) = \sum_{k=2}^{S} p_k(t-1) \cdot a_{ks} \tag{3}$$

where a_{ks} is the probability of transition from state k to state s, with the initialization $p_k(1)=a_{1k}$. It is assumed that the first and the last state of the HMM are non-emitting ones.

D. Calculating model uncertainty

Once the distribution evolutions are built, model uncertainty for models i and j for a given feature is calculated by summing over the overlapping of the model distributions produced in each frame. Hence, the model uncertainty between models i and j for feature k is given by (4):

$$C_k^{(i,j)} = \sum_{t=1}^{T} C_{kt}^{(i,j)} \tag{4}$$

where T is the number of frames for which model uncertainty is calculated, and $C_{kt}^{(i,j)}$ is calculated as the overlapping area of the two distributions (Fig. 2). The curves presented in this plot are assumed to be the distributions $d_t^{(i)}(x)$ calculated with (1). $P(Dj|Mi)$ is the probability of deciding erroneously that model j is detected instead of model i, i.e., the probability of taking model j for model i . Hence, $C_{kt}^{(i,j)} = P(Dj|Mi)$. By defining $h(x)$ as:

$$h(x) = \begin{cases} d_t^{(i)}(x), d_t^{(i)}(x) < d_t^{(j)}(x) \\ d_t^{(j)}(x), d_t^{(j)}(x) < d_t^{(i)}(x) \end{cases} \tag{5}$$

$C_{kt}^{(i,j)}$ can be calculated as the numeric integration of h(x):

$$C_{kt}^{(i,j)} = \int_{-\infty}^{\infty} h(x)dx \cong \int_{a}^{b} h(x)dx \tag{6}$$

In our implementation, the integration is calculated with the trapezoid rule and these values are used: a=-100, b=100, whereas the step e=0.01. This operation is made offline, consequently, a, b and e can be chosen so that the error be minimal. It must be noted that T in (4) will be empircally chosen. The evolution of a distribution lasts to infinity, but due to the exponential decrease of $p_s(t)$ (see (3)), it "fades" after a given number of frames. A reasonable value for T is 50, therefore, considering that for a common value for frame duration of 10 ms, 50 frames are equivalent to 500 ms.

III. Feature Weighting/Removal Algorithms

Section 2 has provided the model uncertainty definition and the way it is calculated. This section describes how to use this measure in order to identify the weights for features or the features to be removed. The removal of features can be viewed

TABLE I. MODEL UNCERTAINTY OF ANY PAIR OF MODELS FOR A GIVEN FEATURE *K*

Model uncertainty for feature *k*	Model 1	Model 2	...	Model *M*
Model 1	$C_k^{(1,1)}$	$C_k^{(1,2)}$	...	$C_k^{(1,M)}$
Model 2	$\mathrm{C_k}^{(2,1)}$	$C_k^{(2,2)}$	...	$C_k^{(2,1)}$
...	...	...	...	...
Model *M*	$C_k^{(M,1)}$	$C_k^{(M,2)}$	...	$C_k^{(M,M)}$

Note that *Ck(i,i)*=1.

as the result of applying weights with values 0 and 1. Model uncertainty between every two models ($C_k^{(i,j)}$) is calculated for each feature. This means that if there are *M* models and *N* features, *N* tables *MxM* are obtained with values of model uncertainty as shown in Table I. Table I is the classical confusion matrix if the decision were made only by feature *k*.

Feature *k* might not discriminate well between models *i* and *j*, but it might help discriminating model *i* from model *l*. However, model *i* is not confused in the same way with all the models. By taking into account these facts, we can give greater weights to the feature that help model *i* to discriminate from the models it is more often confused with. A measure of how much two models are confused can be determined from both the distributions and the recognition results (in terms of WER).

A. Computing model uncertainty and weights from distributions

Section 2.1 shown how model uncertainty represents a volume obtained by the overlapping of distributions, but it is the model uncertainty produced by only one feature. The overall model uncertainty produced by all features represents a volume in the *N*+1- dimensional space (*N* features + time) and it is calculated with (7):

$$C^{(i,j)} = \prod_{k=1}^{N} C_k^{(i,j)} \tag{7}$$

The overall model uncertainty is obtained as a product of all the model uncertainties produced by each feature because it is considered that the features are independent. This is in line with the decoding process, where the features are also considered independent.

This way, another confusion matrix as in Table I is obtained, this time containing the overall model uncertainty.

1) Algorithm 1

The first method consists in weighting features by keeping the overall model uncertainty at a minimal level. It calculates the average model uncertainty of model *i* with all other models for every feature *k*:

$$C_k^{(i)} = \frac{1}{M}\sum_{j=1}^{M} C_k^{(i,j)} \tag{8}$$

Since features with smaller model uncertainty must have greater weight, the weight must be chosen as a monotonically decreasing function of model uncertainty. One way of achieving this is by calculating the weight as the inverse of model uncertainty:

$$w_k^{(i)} = \frac{(C_k^{(i)})^{-1}}{\sum_{k=1}^{N}(C_k^{(i)})^{-1}} \tag{9}$$

The denominator here is used for normalization. $w_k^{(i)}$ are the weights for feature *k* for model *i*. Even though model *i* might be very different from some models and never be confused with them, it might be similar to some other models and thus can often be confused with them. From this point of view, the average model uncertainty in (8) is calculated by summing over the values from the *m* (instead of *M*) models that model *i* is most confused with (i.e. *m* represents the number of models a given model is mostly confused with). *m* is a tuning parameter and may be different for different models, but in our tests, we have used a fixed value for it.

Other monotonically decreasing functions has been attempted instead of the one used in (9). One of them, for example, is the negative exponential. However, none of them has given better results.

2) Algorithm 2

This algorithm is used only when feature removal is desired. It consists in these steps:

```
1. Initialization
    Set the number $K$ of features to be removed
    Empty the pool of removed features

For i=1:$M$  //for each model
    For n=1:$K$ //for each feature to be removed
        2. Find max($C^{(i,j)}$) and the corresponding index $j$.
        3. Find max($C_k^{(i,j)}$) and the corresponding index $k$ for $j$
           calculated at step 2
        4. Introduce $k$ in the pool of removed features
        5. Recalculate $C^{(i,j)}$ with (7) by excluding (not
           multiplying) the terms from the pool of removed
           features
```

The purpose of the algorithm is to remove the least relevant features. Here, the least relevant features are considered the ones that introduce the highest amount of model uncertainty (steps 2 and 3 of the algorithm). After each iteration, the average model uncertainty is recalculated by excluding the already removed features (step 5 of the algorithm).

B. Calculation of confusions and weights by using the recognition results

This method does not replace **Algorithm 1**, but it completes it. The basic idea is to get an indication of how often two models are confused with each other during the recognition process, by building a confusion matrix. This is realized by analyzing the recognition results with the training and testing databases and counting how many times model *j* has been recognized instead of model *i*. It must be observed that the number of confusions calculated in this way depends on the data chosen for training and on the context the speech unit is found in (i.e. a phoneme within a word). This information is precious because many times confusions calculated in this way do not match perfectly the confusions calculated from distributions.

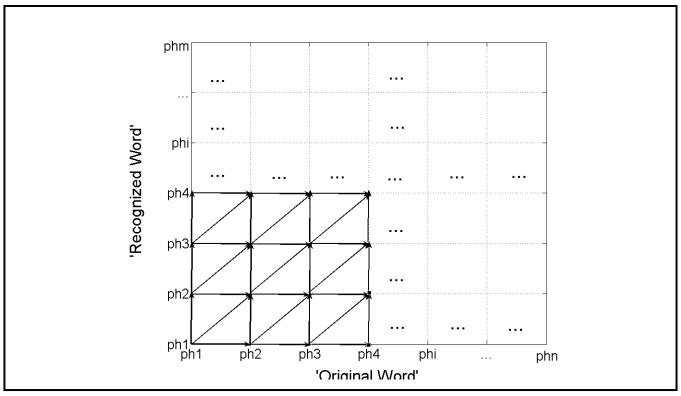

Figure 5. DTW algorithm used for aligning the recognized word to the original one

In our experiments, phonemes are used as models, hence, in order to count the confusions, we have to align the recognized word to the original one and see which phoneme is recognized erroneously. The errors in ASR, at a phonemic level, consist in substitutions, insertions and deletions. These errors are counted automatically by applying a Dynamic Time Warping (DTW) algorithm as shown, in Fig. 5. Phonemes from the vertical axis belong to the recognized word, whereas phonemes from the horizontal axis belong to the original word uttered in the speech recording. The algorithm searches for the path with the lowest cost among the paths that originates from the point (ph1, ph1) and ends to the point (phn, phm). The costs are presented in Table II.

The costs are chosen so that they can cover all the types of ASR errors (substitutions, insertions and deletions). If there are no errors, the optimal path is the diagonal and its cost is equal to 0. Horizontal transitions correspond to deletions. Vertical transitions, which correspond to insertions, have lower costs than the horizontal transitions because this comes down to two phonemes being recognized instead of one. Many times the recognized phonemes are the correct phonemes and another one, or the correct phoneme twice.

After finding the path with the lowest cost, which means that the best alignment is found, its nodes are analyzed: if phonemes (from the horizontal and vertical axes) do not match, then the E_{ij} counter is incremented, where i is the phoneme from the horizontal axis (the correct one) and j is the phoneme from the vertical axis (the recognized one). These counters are normalized by the total number of occurrences for phoneme i, i.e. n_i:

$$e_{ij} = \frac{E_{ij}}{n_i} \tag{10}$$

After analyzing all the words, Table III is filled in with the e_{ij} values.

The algorithm used is the same as **Algorithm 1**, but in (8), instead of using $C_k^{(i,j)}$ the geometric mean $\sqrt{C_k^{(i,j)} e_{ij}}$ is used.

TABLE II. COSTS OF THE DTW ALGORITHM

	Cost
Node with phoneme match	0
Node with phoneme mismatch	4
Diagonal transition	0
Horizontal transition	2
Vertical transition	1

TABLE III. CONFUSION MATRIX YIELDED BY THE RECOGNITION RESULTS

model \ confused with	Model 1	Model 2	...	Model M
Model 1	e_{11}	e_{12}	...	e_{1M}
Model 2	e_{21}	e_{22}	...	e_{2M}
...	...	...	...	...
Model M	e_{M1}	e_{M2}	...	e_{MM}

In the Section 4, it will be referred to as **Algorithm 3**. **Algorithm 1** differs from **Algorithm 3** because, when calculating weights, it uses only the trained distributions of the models. In **Algorithm 3**, confusions obtained from the recognition results also contribute to the weight calculation. The geometric mean is used because it gives better results than the arithmetic mean. It is expected to have similar values for $C_k^{(i,j)}$ and e_{ij}, but in the recognition results, some models are confused with only one model or mostly with one model. This does not always reflect the reality, but it is because of the database and its limited number of contexts for a phoneme. The geometric mean can reduce this effect because, compared to arithmetic mean, it is always closer to the smallest scalar.

IV. EXPERIMENTAL RESULTS

The experiments are made on an ASR system trained with a database of 10 000 distinct words in the Romanian language. The words were chosen so that they cover all the syllables of the Romanian language. Five different speakers have uttered these words in the same laboratory conditions, resulting in a database of 50 000 word tokens. A set of 5000 recordings (10% of all recordings) is used for testing (the same set of 1000 words was chosen for each of the 5 speakers), whereas the rest of the 45 000 words are used for training. Labeling is made at a word level. Since each word is recorded in a separate file, the label file contains the following sequence: *sil* - <word> - *sil*, where *sil* stands for silence. Silence is also modeled as an independent HMM. The HMM chosen for modeling has 12 states, where the first and the last state are not emissive. Transitions are allowed only from left to right and from a given state j the model can transit only to the states j, j+1 and j+2. The output *pdf* is a mixture of 3 Gaussian distribution functions. As voice features, the MFCCs (12 coefficients) including energy are used (13 coefficients), along with their first-order derivatives, for a total of 26 features.

The HTK Toolkit [18] has been used for both training and recognition. The training process is performed classically by using the Baum-Welch algorithm, while for decoding the Token Passing algorithm is used as described in [17]. Recognition tests are run in three modes:

- Grammar-Free Mode (GFM) – No restriction is imposed on the number of words or on their order.
- Simple Grammar Mode (SGM) – It originates from the assumption that only one word can be uttered at a time (the speech files contain only one word). So, only the succession *sil* - <word> - *sil* is allowed.
- Reduced-Dictionary Simple Grammar Mode (RDSGM) – The grammar used is the same as for SGM, but the dictionary is reduced to 1000 words (instead of the 10 000 words used in GFM and SGM).

Grammar is used because in human-machine interaction applications based on singular words, many errors are triggered by the recognizer's inability to perfectly split a compound word into two shorter words. Applying a simple grammar as in SMG prevents these errors, which can also be avoided by introducing an insertion penalty [18]. The reduction of the dictionary is justified by the existence of simple applications, such as menu navigation on mobile devices, where a large dictionary is not needed. In Table IV, we show the baseline that will be used in the analysis of the results.

Recognition rate is measured as the ratio of the number of word recognized correctly and the total number of words [17].

Two directions for tests are developed; first, weights are used in order to obtain higher recognition rates, second, the removal of features (weights of 1 and 0) is utilized in order to decrease the number of calculations, while trying to maintain the same recognition rate.

The results obtained by applying **Algorithm 1** are presented in Table V, where *m* represents the number of models a given model is mostly confused with.

The results show that there is an optimal value for *m*. This is as expected, because weights are calculated from the model uncertainties of one model with models it is more often confused with. For small values of *m* the system will continue to confuse one model with the similar models not included within *m*, whereas for great values of *m* the system is not oriented anymore to discriminate one model from the models it is more often confused with. For GFM the optimal value for *m* is 3, while for SGM and RDSGM this value is 2, which means that, on average, one model is mostly confused with other two or three models. In these conditions, it is indicated that the weights of a model be calculated based on the confusions with the 2-3 models it is more often confused with. The relative improvement is about 4.2%, which is encouraging, because by refining the function that calculates weights, better results can be obtained.

The same algorithm is also used for features removal. For each testing mode, the number *m* which yields better results, is taken from Table V. Then, different results (Table VI) are obtained for different numbers of removed features (*K*).

The results obtained by applying **Algorithm 2** are presented in Table VII.

Algorithm 2 gives better results than **Algorithm 1**. They both show that for these models all the features contribute in discriminating among models (because the scores decrease when removing more features), but it confirms that not all the features have the same importance, because the decrease of scores is very small. For some applications implemented on devices with limited processing power a compromise can be made. As the number of features used is 26, removing 3 features means an absolute reduction in the number of calculations of costs by 11.5%, while removing 5 features implies a reduction of the number of calculations by 19.2%.

The above tests are obtained by analyzing the trained models. As explained in Section 3.2, **Algorithm 3** takes into account an indication from the recognition results about how often two models are confused. The obtained results for both weighting and features removal are presented in Tables VIII and IX.

TABLE IV. ERROR RATES WITHOUT USING WEIGHTS

Recognition mode	WER [%]
GFM	15.5
SGM	8.22
RDSGM	2.47

TABLE V. ERROR RATES WHEN USING **ALGORITHM 1**

	WER [%]		
m	GFM	SGM	RDSGM
1	15.56	8.19	2.47
2	15.18	7.77	2.4
3	14.95	7.82	2.44
4	15.08	8.35	2.49
5	15.68	8.44	2.53

TABLE VI. ERROR RATES WHEN USING **ALGORITHM 1** FOR FEATURES REMOVAL

	WER [%]		
Number (*K*) of removed features	GFM (*m*=3)	SGM (*m*=2)	RDSGM (*m*=2)
1	15.8	8.3	2.47
2	15.96	8.38	2.51
3	16.58	8.88	2.82
4	18.88	9.4	3.11
5	20.12	10.36	3.39

TABLE VII. ERROR RATES WHEN USING **ALGORITHM 2** FOR FEATURES REMOVAL

	WER [%]		
Number (*K*) of removed features	GFM	SGM	RDSGM
1	15.6	8.22	2.47
2	15.94	8.38	2.49
3	16.08	8.56	2.58
4	16.22	9.1	2.74
5	17.47	10.3	3.02

TABLE VIII. ERROR RATES WHEN USING **ALGORITHM 3**

	WER [%]		
m	GFM	SGM	RDSGM
1	15.52	8.13	2.47
2	15.03	7.69	2.38
3	14.87	7.79	2.47
4	15.04	8.4	2.47
5	15.56	8.48	2.57

TABLE IX. ERROR RATES WHEN USING **ALGORITHM 3** FOR FEATURES REMOVAL

	WER [%]		
Number (K) of removed features	GFM (m=3)	SGM (m=2)	RDSGM (m=2)
1	15.6	8.3	2.47
2	16.02	8.38	2.51
3	16.12	8.66	2.58
4	18.02	9.66	2.97
5	19.5	10.4	3.39

These results show that **Algorithm 3** gives slightly better results for weighting coefficients. This derives from the fact that, unlike **Algorithm 1**, it has supplementary information provided from the recognition results. Many times the phoneme confusion is influenced by the context (for example, the position of a phoneme in a word), but confusions calculated only from the trained models (as in **Algorithm 1**) do not have contextual information. **Algorithm 3**, instead, by using this information, calculates more accurate weights.

Regarding the results for features removal, they are better than **Algorithm 1**, but slightly worse than **Algorithm 2**. Results may be better if a way of combining **Algorithm 2** and **Algorithm 3** is found.

Results show that the contribution of features in discriminating among models can be optimized by introducing weights that are calculated as presented in Sections 3.1 and 3.2. WER decreases with both **Algorithm 1** and **Algorithm 3**. Even though the improvement is small, it is encouraging because the algorithms can be improved, for example, by using another weighting function than (9), or by calculating uncertainty at a state level instead of a model level. In the case of features removal, WER increases with all algorithms, but this increase is very small. In applications were the processing power is limited (like in portable devices), the decrease in recognition can be traded for an improvement in efficiency.

V. CONCLUSIONS AND FUTURE WORK

The goal of this paper is the optimization of the recognition process in terms of quality and reduction of calculations. For this purpose we have defined *model uncertainty* as a measure of overlapping between models. Feature weights are calculated, for each model, based on model uncertainty. Weights are calculated from model uncertainty with a monotonically decreasing function. Model uncertainty is calculated from the distributions of the trained models but it is also calculated from recognition results.

Two directions of tests have been explored: using weights for increasing the recognition rate and removing the features with the lowest weights, so that the number of calculations of scores is reduced. Experiments have shown that not all the features have the same contribution in discriminating models, and the use of weights can give better results. Moreover, irrelevant features (those with lower weights) can be removed and the system performance is not significantly degraded.

The results can be improved by refining the function which calculates weights from confusions. The configuring parameter m, which represents the number of models a given model is mostly confused with, has a fixed value. However, the optimal value for m is different for each model, therefore a study will be made in order to determine a customized value of m for each model.

The last improvement consists in calculating confusions among states of HMM instead of models. It is expected that a *"per state"* implementation of the solution will give better results, because in calculating the costs per state, the comparison between states is more direct than between models.

ACKNOWLEDGMENT

The research reported in this paper was funded by the Romanian Government, under the National Research Authority CNCSIS grant "IDEI" no. 114/2007, code ID_930 and by the Sectoral Operational Program Human Resources Development 2007-2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Agreement POSDRU/6/1.5/S/16.

REFERENCES

[1] J. A. Arrowood, "Using observation uncertainty for robust speech recognition," Ph. D. Thesis, Georgia Institute of Technology, Atlanta, GA, 2003.

[2] J. Hamaker, J. Picone, A. Ganapathiraju, "A Sparse Modeling Approach to Speech Recognition Based on Relevance Vector Machines," Proc. ICSLP, vol. 2, pp. 1001-1004, 2002.

[3] J. Bi, T. Zhang, "Support vector classification with input data uncertainty," Proc. NIPS, pp. 161-168, 2004.

[4] A. D. Subramaniam, B. D. Rao, "PDF Optimized Parametric Vector Quantization of Speech Line Spectral Frequencies," IEEE Trans. SAP, Issue 2, pp. 130-142, 2003.

[5] E. Bocchieri, B.Mak, "Subspace Distribution clustering Hidden Markov Model," IEEE Trans. ASSP, Vol. 9, pp. 264-275, 2001.

[6] S. Kanthak, K. Schutz, H. Ney, "Using SIMD instructions for fast likelihood calculation in LVCSR," Proc. ICASSP, vol. 3, pp. 1531-1534, 2000.

[7] M. Vasilache, "Speech recognition using HMMs with quantized parameters," Proc. ICSLP, pp. 441–444, 2000.

[8] D. Huggins-Daines, et al., "PocketSphinx: a free, real-time continuous speech recognition system for handheld devices," Proc. ICASSP, pp. 185–188, 2006.

[9] L. Hetherington, "PocketSUMMIT: Smallfootprint continuous speech recognition," Proc. INTERSPEECH, pp. 1465–1468, 2007.

[10] M. Novak, "Towards large vocabulary ASR on embedded platforms," Proc. INTERSPEECH, pp. 2309-2312, 2004.

[11] J. Droppo, A. Acero, L. Deng, "Uncertainty decoding with splice for noise robust speech recognition," Proc. ICASSP, pp. 57-60, 2002.

[12] H.Liao, M.J.F. Gales, 2005. "Joint uncertainty decoding for noise robust speech recognition," Proc. INTERSPEECH, pp.3129-3132, 2005.

[13] Y. Liu, P. Fung: "Acoustic and phonetic confusions in accented speech recognition," Proc. INTERSPEECH, pp. 3033-3036, 2005.

[14] B. Meyer, M. Wächter, T. Brand, B. Kollmeier, "Phoneme Confusions in Human and Automatic Speech Recognition," Proc. INTERSPEECH, pp. 1485-1488, 2007.

[15] G. Bouselmi, D. Fohr, I. Illina, J.P. Haton, "Fully Automated Non-Native Speech Recognition Using Confusion-Based Acoustic Model Integration," Proc. EUROSPEECH/ INTERSPEECH, pp. 1369-1372, 2005.

[16] Y. Chen, C. Wan, L. Lee, "Entropy-based feature parameter weighting for robust speech recognition," in Proc. ICASSP, pp. 41-44, 2006.

[17] H.-G. Hirsch, D. Pearce, "The AURORA Experimental Framework for the Performance Evaluation of Speech Recognition Systems under Noisy Conditions," Proc. ISCA ITRW ASR2000, pp. 29-32, 2000.

[18] G. Everman, et al., “The HTK Book”, Version 3.0, Cambridge University, Engineering Department, 2005.

[19] S.J. Young, N.H. Russell, J.H.S. Thornton, “Token passing: a simple conceptual model for connected speech recognition systems”, Technical Report, Cambridge University: Engineering Department, 1989.