# EFFICIENT NN-BASED SEARCH SPACE REDUCTION IN A LARGE VOCABULARY SPEECH RECOGNITION SYSTEM

**J. Macías-Guarasa, J. Ferreiros, J.M. Montero, R. Córdoba and Á. Olbés**
**Speech Technology Group. Department of Electronic Engineering**
**ETSI de Telecomunicación. Universidad Politécnica de Madrid. Spain**
macias@die.upm.es

## ABSTRACT

In very large vocabulary speech recognition systems using the hypothesis-verification paradigm, the verification stage is usually the most time consuming. State of the art systems combine fixed size hypothesized search spaces with advanced pruning techniques. In this paper we propose a novel strategy to dynamically calculate the hypothesized search space, using neural networks as the estimation module and designing the input feature set with a careful greedy-based selection approach. The main achievement has been a statistically significant relative decrease in error rate of 33.53%, while getting a relative decrease in average computational demands of up to 19.40%**.**

**KEYWORDS:** Speech recognition, neural networks, search space reduction, hypothesis-verification systems, greedy-based feature set selection.

## 1. INTRODUCTION

Computational demands are one of the main factors to take into account when designing systems supposed to operate in real-time, especially when talking about public information services using the telephone network. Telephone information service providers are demanding systems and algorithms that allow them to increase the number of active recognizers to run in dedicated hardware, to be able to significantly decrease production costs.

According to this scenario, state of the art systems are usually based in some form of *progressive search* [1], whereby successively more detailed (and computationally expensive) knowledge sources are brought to bear on the recognition search as the hypothesis space is narrowed down. This approach is a generalization of the hypothesis-verification paradigm, with several cascaded stages. In the simplest case the first stage (hypothesis), a *rough analysis* module with low computational demands, face the whole search space of the task, and select a subset of this search space to be fed to the second stage (*detailed analysis* module, verification), much more demanding in computational resources and more able to accurately decode the input speech. For the whole system to be successful, the rough analysis module must ensure that the selected subset of the search space contains the right hypothesis with high probability, so as not to degrade the overall performance.

In hypothesis-verification systems, the main concern is reducing the hypothesized search space as much as possible, and this is not an easy task, especially when low detailed acoustic models are used in the preselection stage. Traditionally, these systems use a fixed size hypothesized search space, estimated according to the results obtained during system development so that a minimum error rate is achieved. Under these constraints, most of the research work aimed at lowering computational requirements has been centered in search space pruning techniques, usually based in beam search techniques [2].

Our proposal is focused in the hypothesis stage: instead of only relying in static or dynamic pruning techniques in the verification module, we want to design a procedure so that the hypothesized search space varies in size, different for every utterance, depending on any knowin-advance system parameter. If we lower the *average* hypothesized search space size while keeping the error rate performance, the computational demands of the overall system would be lower. Thus, the key factor to evaluate the effectiveness of different methods is calculating the reduction in average hypothesized search space size (which we will refer to as *average effort*) while keeping the required error rate. In addition to that, if the neural network estimation is accurate enough, we could even get improvements in the system error rate, and this actually happens in the experiments described below.

## 2. SYSTEM OVERVIEW

The general architecture we are working on is shown in Figure 1, in which an estimator module is in charge of deciding the size of the hypothesized search space, to be passed to the detailed analysis module, using for that purpose a certain set of features extracted from the feature extraction and rough analysis processes. This estimator module will be the one in which we will use NNs as the estimation strategy. The main hypothesis generator modules are fully described in [3], and to summarize, the current implementation of the hypothesis module follows a bottom-up, two stage strategy (a phonetic string is first generated and then matched against a tree-structured dictionary, using dynamic programming algorithms [4] and [5]).

In general, and given the proposed task, it is clear that the computational requirements are closely related to two different factors: modeling complexity and search effort in the hypothesis and verification algorithms. In this paper, we will focus our description to the work in the preselection (hypothesis generation) system, although additional reductions in computational demands are achieved in the verification module through the use of beam pruning techniques. Our target objective will be achieving a maximum error rate, which should be lower than 2% in all cases, so as not to limit the final recognition accuracy achieved by the verification stage.

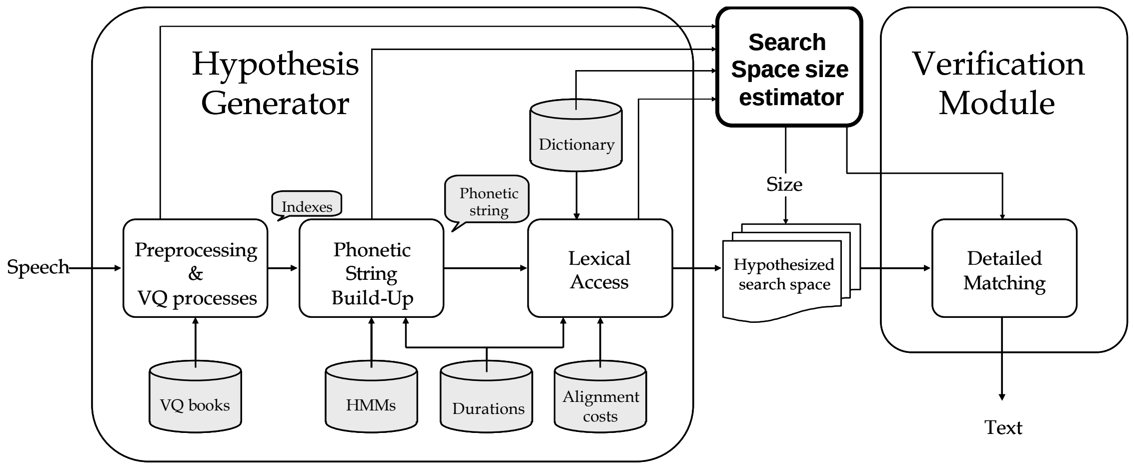

*Figure 1. System architecture*

## 3. EXPERIMENTAL SETUP

Experiments have been carried out using part of VESTEL, a realistic isolated word telephone speech database [6], captured using the Spanish Public Switched Telephone Network and composed of 9,720 utterances. We have used the *leave-one-out* method with ten subgroups in order to increase the statistical significance of the results. For every subgroup, 80% of the data is used for training, 10% for validation and tuning, and the remaining 10% for the final evaluation. The validation subset is used to make decisions regarding the best feature selections, the optimal number of iterations, etc. All the training-validation-testing procedure is repeated for each of the 10 *leave-one-out* subgroups and the results are finally averaged.

The real-world application task was designed for the research and development division of the Spanish Telephone Company (Telefónica I+D) around the *white pages* idea, so that the dictionary used in this work is composed of 10,000 words, the most probable names and surnames in Spanish. The experiments will be carried out in the context of a large vocabulary isolated word recognition system. In this case, the hypothesis module will generate a *preselection list* composed of the most probable words (candidates) given the input speech utterance. The **preselection list length** (***PLL*** from now on) used, on the average, would give us the *average effort* for the task.

## 4. BASELINE SYSTEM

The baseline experiment uses fixed *PLL*s, which is equivalent to a fixed search space size. For its evaluation, we calculated the *inclusion error rate* achieved for every possible length of the preselection list. The inclusion error rate is obtained assuming a recognized word is within the first *N* candidates (*N* equals the *PLL*) proposed by the hypothesis module. In general, and given the unequal performance of recognition systems depending on the word to be recognized, that *fixed length* must be assigned a large value, leading to a large *average effort*, a large average *wasted effort* and to computational requirements higher than desired. Actual system performance measurements showed that the *wasted effort* almost equals the required *average effort*, so that great improvements could be achieved.

In our system, we obtained 2% error rate for a fixed *PLL* of around 10% of dictionary size, which is 1000 candidates. Taking into account this result and previous experiments, we established the baseline system as the one that used exactly 1000 candidates for the fixed *PLL*, which lead to an inclusion error rate of 1.72%.

So, our target will be achieving at least the same performance (1.72% error rate) while, reducing the average *PLL* (which equals to the average hypothesized search space size, thus lowering the computational demands for the whole system), as we will use variable *PLL*s estimated using a neural network (NN).

## 5. NNS AS HYPOTHESYS SEARCH SPACE SIZE ESTIMATORS

In all our experiments we will use a multi layer perceptron [8], with a single hidden layer and sigmoids as the activation functions. In order to increase the generalization capabilities of the NN, we kept the number of neurons as low as possible, leading to relatively simple topologies with less than 600 weights to train.

### 5.1. Parameter inventory and input coding

In our case we have created a wide spectrum of possibilities regarding the available feature set: We designed an inventory of 56 features that can be classified in three broad classes:

- Direct parameters: Obtained from the characteristics of the acoustic utterance or the preselection process: number of frames, phonetic string length, acoustic search score, lexical access costs, etc.
- Derived parameters: Calculated from the previous ones applying different types of normalization schemes (dividing by number of frames, phonetic string length, etc.)
- Lexical Access Statistical Parameters: Averages and standard deviations calculated over the lexical access costs distribution, for different *PLL*s.

The input coding schemes we tested include both single and multiple inputs per parameter:

- For parameters coded in a single input, the alternatives were: No coding (raw data input), linearly scaling the full parameter range between a minimum and maximum value of the input neuron, standard *z-score* normalization, with optional data clipping to some predefined values, proportional to the standard deviation of the training data set.
- For parameters coded in multiple inputs:
  - Using a uniformly distributed linear mapping function: dividing the full parameter range by the number of inputs, so that we activate the input corresponding to the range in which the parameter value is located.
  - Using a non-uniform mapping function: considering the distribution of the parameter values in the training database, the segments assigned to each input are chosen so that the number of activations is equalized for each parameter.

In addition to that, we tested different number of input units per parameter and different values to encode every input activation. We tested all the different coding schemes using a subset of the training database and we established that standard *z-score* normalization for the input features achieved the best results [7].

## 5.2. Output coding and NN post-processing

Our network is aimed at estimating a certain *PLL*, given the input parameters. For coding the activations of the output neurons we could use the same strategies we discussed above for input parameter coding. In this case we are interested in multiple output neurons, as they could encode *PLL* values in a better way. In output coding, however, using a uniformly distributed linear mapping function lead to very bad results, as only the first few neurons are activated during training, as most utterances are recognized for the first few candidates in the preselection list.

We evaluated all the possible output coding strategies and we established that the best results where obtained with the following setup [7]:

- Every output neuron *k* is defined to represent a different *PLL* range (*PLL*s from *lowerSegmentLength(k)* to *upperSegmentLength(k)*), leading to the task formulated as a classification problem in which the NN should decide which is the most likely output neuron to be activated.
- The *PLL* ranges that every output neuron represents are trained with a criterion that aims to get, when possible, a uniform number of training samples for all of them, in order to avoid data sparseness during training.

The NN output values are finally post-processed to obtain the final *PLL*. The idea is further increasing the proposed length, so that mismatches between the training and testing sets are compensated to a certain extent. Different alternatives where tested:

- The output neuron with the higher activation value decides the *PLL* to be used (the upper limit of the *PLL* range associated to the winning neuron). If *act(k)* is the activation value for the *k* output neuron:

$$PLL = upperSegmentLength(g), \quad g = \underset{0<k\leq numOutputNeurons}{\arg\max} \left[act(k)\right] \tag{1}$$

- The *PLL* is calculated as a linear combination of output neuron activations multiplied by the upper limit of the *PLL* range associated to each output neuron. The rationale for this approach is based on the fact that, given certain premises, NN outputs can be interpreted as class posterior probabilities [8], so that all output neurons have something to say regarding the estimated *PLL*:

$$PLL = \sum_{k=1}^{NumOutputNeurons} upperSegmentLength(k) \cdot act(k) \tag{2}$$

From these basic approaches, two additional mechanisms were tested to increase system robustness: Adding a fixed threshold to the proposed *PLL*. If *PLL** is the final *PLL* to be used:

$$PLL^* = PLL + fixedTrainedThreshold \quad (3)$$

or using a proportional threshold to the proposed *PLL*:

$$PLL^* = PLL \cdot (1 + proportionalTrainedThreshold) \quad (4)$$

Of course those thresholds are also calculated during the training phase, imposing the achievement of a certain inclusion rate. We tested all the NN post processing strategies, and the best one proved to be the one given by equation (2), with the threshold equation (3) [9].

## 5.3. Feature selection

The initial experimentation described in sections 5.1 and 5.2 gave us the experimental scenario to be used in the feature selection process. In order to select the most discriminative features for our task, we used an adapted version of the *greedy* algorithm [10]. Initially, this procedure was planned as follows:

1. The feature set is initialized as empty.
2. In every iteration of the feature selection algorithm, feature ensembles are generated adding every pending feature to the existing feature set.
3. Experiments with variable number of iterations are performed for every feature ensemble.
4. The feature ensemble achieving the highest reduction in preselection error rate for the optimal number of iterations is selected as the new feature set for the next iteration.
5. Continue with step 2 if the preselection error rate decreases.

Initial evaluation showed that the computational complexity of step 3 is huge, so that we simplified the process as follows:

- In steps 3 and 4 we select the 8 feature ensembles which lead to the best results using a training procedure with the optimal number of iterations found *in the previous iteration*.
- Before step 5 we carry out experiments to determine the optimal number of iterations for each of the 8 best ensembles and we finally select the ensemble with the highest reduction in preselection error rate.

With this approach, a set of 4 features was selected as the optimum. The most discriminative features are related to the standard deviation of the lexical access costs, the normalized acoustic score from the phonetic string build up module and the phonetic string length.

## 6. EXPERIMENTAL RESULTS

As discussed above, the NN-based system will generate a different *PLL* for every utterance. Inclusion error rates are calculated according to this approach and can be directly compared to the baseline system inclusion error rate (1.72%). On the other hand, computational requirements in the NN-based system are measured computing the *average* estimated *PLL*, and will be compared with the fixed size of 1000 candidates in the baseline system.

Obviously we still need relative quality measurements, so that we will also calculate relative error rate and average effort increments (Δ) when using the NN-based system. Table 1 shows the final results (along with 95% confidence intervals for the error rate figures).

| | *Inclusión error rate* | *Average effort* | *D error (% relative)* | *D effort (% relative)* |
|---|---|---|---|---|
| ***Fixed list length system*** | 1.72% ± 0.21% | 1000 | - | - |
| ***NN based system*** | 1.14% ± 0.26% | 806 | -33.53% | -19.40% |

Table 1. Experimental results

So, the best NN based system achieves a 33.53% reduction in error rate, while also reducing the average computational effort in almost 20%. In addition to that, the differences are statistically significant (confidence intervals do not overlap).

The computational impact of the NN calculations is negligible when compared to the overall runtime of the preselection stage (under 0.01% of the total runtime).

## 7. CONCLUSIONS AND FUTURE WORK

The main conclusion of the presented paper is that the use of neural networks as the estimators of the hypothesis search space size has proven to be an excellent alternative to the traditional approach of using fixed sizes, clearly outperforming it (33% reduction in error rate and 20% reduction in computational demands) and with statistically significant results.

We also presented a carefully designed experimental methodology, using a greedy-based strategy for feature selection and an optimal experimental setup regarding input and output coding, along with a NN output post-processing system that relies in the interpretation of the NN outputs as being class posterior probabilities.

Given the good results obtained in our classification networks, we have started a study on the estimation of word confidence measures, to allow assessing the reliability of the recognition systems used. Initial results are really encouraging, as we are outperforming traditional methods using parameters related to acoustic scores with some of the ones proposed in this paper.

## 8. REFERENCES

1. J.L. Gauvain and L. Lamel, “Large-vocabulary continuous speech recognition: advances and applications”. *Proceedings of the IEEE*, Volume: 88, Issue: 8, pp. 1181-1200. August 2000.
2. S. Ortmanns, H. Ney and A. Eiden, "Language-Model Look-Ahead for Large Vocabulary Speech Recognition", Proc. Int. Conf. on Spoken Language Processing, vol. 4, Philadelphia, PA, USA, 1996, pp. 2095-2098.
3. J. Macías-Guarasa, A. Gallardo, J. Ferreiros, J.M. Pardo and L. Villarrubia, "Initial Evaluation of a Preselection Module for a Flexible Large Vocabulary Speech Recognition System in Telephone Environment". Proc. Int. Conf. on Spoken Language Processing, Philadelphia, PA, USA, 1996, pp. 1343-1346.
4. H. Ney. “The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition”. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, n. 2, 1984, pp. 263-271.
5. L. Fissore, P. Laface, G. Micca and R. Pieraccini, “Lexical Access to Large Vocabularies for Speech Recognition”. *IEEE Transactions on Acoustics, Speech and Signal Processing* vol. 37, n. 8, 1989, pp. 1197-1213.
6. D. Tapias, A. Acero, J. Esteve and J.C. Torrecilla, "The VESTEL Telephone Speech Database". Proc. Int. Conf. on Spoken Language Processing, Yokohama, Japan, 1994, pp. 1343-1346.
7. J. Macías-Guarasa, J. Ferreiros, J. Colás, A. Gallardo-Antolín and J.M. Pardo, “Improved Variable Preselection List Length Estimation Using NNs In A Large Vocabulary Telephone Speech Recognition System”. Proc. Int. Conf. on Spoken Language Processing, Beijing, China, 2000, pp. 823-826.
8. C.M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. pp. 116-161 and pp. 245-247.
9. J. Macías-Guarasa, “Architectures and Methods in Large Vocabulary Speech Recognition Systems.” PhD. Thesis. Universidad Politécnica de Madrid, 2001.
10. J. Kittler, ”Feature set search algorithms” in *Pattern Recognition and Signal Processing*, C.H. Chen, Ed., Sijthoff and Noordhoff, The Netherlands, 1978, pp. 41-60.