# AUTOMATIC SET-UP FOR SPEECH RECOGNITION ENGINES BASED ON MERIT OPTIMIZATION

*G. Hernández-Ábrego, X. Menéndez-Pidal*

Spoken Language Technology
Sony Electronics, Inc. NSCA
3300 Zanker Rd, San Jose CA 95134
gustavo@slt.sel.sony.com

*T. Kemp*

ATCS-MMI
Sony International Europe
Hedelfinger Str. 61
70327 Stuttgart, Germany

*K. Minamino, H. Lucke*

Digital Creatures Lab. Gp1
Sony Corporation
6-7-35, Kitashinagawa
Shinagawa-ku, Tokyo, Japan

**ABSTRACT**

We propose an automatic method to set-up the several parameters that define the behavior and performance of a typical speech recognition engine. Such parameters include weights and beam widths among others. Our method is based on the definition of a merit function. Here, merit is understood as an intuitive notion of recognition performance based on both recognition accuracy and computation time. A convenient definition of merit allows to apply an optimization procedure to define a convenient set-up for the recognizer with little human intervention. The method is applied to adjust the recognition parameters of two different LVCSR applications, one in American English and another in Japanese.

## 1. INTRODUCTION

For use of speech recognition in systems and devices used in everyday life, accurate recognition performance is crucial. However, for a practical application, issues about real-time performance and memory usage cannot be ignored. Accurate acoustic and language models are a pre-requisite for high recognition accuracy, however they usually also imply significant computational resources, which are not always available. For a practical system, it is therefore necessary to limit the use of such resources while maintaining accuracy. For this purpose, a number of approximations, look-ahead techniques and other compromises (such as limitation of search space, introduction of flooring values, etc.) generally are employed. These techniques do impact the recognition performance when used carelessly. Careful tuning of the parameters controlling these techniques, such as beam-widths, look-ahead depths, parameters controlling approximate calculations and the like, is required. This usually results in a cumbersome process which requires many experiments to be done under the guidance of experienced users, making the system performance dependent on personal knowledge and experience. What makes things worse is that many of these parameters appear to be dependent on the actual acoustic and language models, implying that when these change, the tuning process need to be repeated for optimal performance.

Whether this tuning is done by ”brute force”, meaning testing a huge number of parameter settings, tabulating the results and finding the optimum, or by a systematic parameter optimization such as Powell’s algorithm, which is employed e.g. by Seymore for language model weight optimization [1], in either case a huge number of experiments under human supervision is required. Clearly the number of experiments required increases as more parameters for system optimization are introduced.

In this paper we propose an innovative method to adjust the recognition parameters through optimization with little input from the user. Optimization is based on defining the *merit* of each recognition experiment. Merit is a subjective notion suitable to be evaluated in a natural way through soft computation based on fuzzy logic. Merit can be understood as a function that depends on the configuration of the recognition system and on its set-up. When used as a cost function, the merit can be the basis for finding the optimal set-up of a given recognition system.

## 2. FUZZY MERIT EVALUATION

In a practical recognition system, overall performance depends on a combination of system attributes such as WER, memory, RTF, etc. which need to be jointly optimized. In this paper we apply the method to just two attributes: WER and RTF. In speech recognition, a trade-off between WER and RTF seems apparent: for less recognition errors, more computation time is needed. For practical purposes, a recognition system yielding high accuracy at high operational speed is very desirable. In the evaluation of speech recognition performance, WER and RTF can be combined to express a joint assessment. This joint measure can be regarded as recognition merit. In the research work reported here, we did not try to address any analytical expression of the merit. Instead, we used the common sense notion stated above to drive our merit definition. Intuitive and subjective knowledge are hard to include into numeric computation unless they are added in a heuristic manner. Fuzzy logic represents an adequate theoretical framework to efficiently handle subjective notions in a formal representation. A fuzzy logic system [2] is based on membership functions and fuzzy rules which, when properly defined, can combine real-life numeric measurements with intuitive knowledge into a logical output result. In an oversimplified description, the basic steps involved in a fuzzy logic system can be summarized as follows:

Fuzzification: maps the input values into fuzzy notions.

Rules and inference: “if . . . then” statements that indicate how the fuzzy notions are combined.

Defuzzification: maps back a fuzzy notion to an output value.

Fuzzy logic can be interpreted as a mapping tool capable of transforming intuition and linguistic operations into a systematic model built from elementary mathematical functions. The operations within a fuzzy logic system can be brought “down to earth” when expressed in basic mathematical terms. The fuzzification

process is nothing more than a mapping of input values into pre-defined fuzzy sets. Each fuzzified input is then passed to a set of fuzzy rules. Rules combine the fuzzy notions through the fuzzy "and" operator that, in this case, is nothing more than the product of them all. Every rule produces an output that later is to be averaged with the rest of rule outputs to build a single and unified fuzzy result. Strictly speaking, defuzzification is not used in our scheme because the merit, a fuzzy notion, is our optimization target.

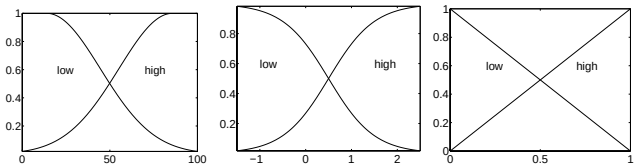
In our system, a fuzzy logic system with three inputs and one output is used. The inputs include the WER and the log RTF. Log values of the computation time are used to restrict the dynamic range of this variable. There is a third input called the "WER_over_RTF" (WOT) value conceived to balance the trade-off between WER and RTF in recognition. For high WOT values, accuracy is more relevant than speed. The actual value of WOT is critical to the entire optimization process since it greatly affects the shape of the cost function and thus changes the point reached during optimization.

In formulae, our fuzzy logic system brings a definition of merit according to:

$$M(\vec{\Theta}) = \frac{\sum_r \left( z(r) \prod_{x \in X} f(x) \right)}{\sum_r \left( \prod_{x \in X} f(x) \right)} \qquad (1)$$

where $r$ is the fuzzy rules index. The fuzzy system inputs, $X = \{W(\vec{\Theta}), \log R(\vec{\Theta}), WOT\}$, are functions of the recognition engine set-up $\vec{\Theta}$: a set of recognition parameters (weights, lengths or ranges) that define the behavior of the system. For fuzzy merit computation, the membership functions $f(x)$ are sigmoidal for $W(\vec{\Theta})$ and $\log R(\vec{\Theta})$, and are linear for $WOT$. The combination of rules is given by the product in (1). Each combination is later weigthed with a constant value $z(r)$ associated to each rule. The system output, merit in this case, is the result of the weighted average of all rule outputs.

The actual implementation of our fuzzy system to measure the merit depends on 1) the definition of the membership functions to fuzzify the inputs and 2) the rules to define the relationship between them. Each input variable in the fuzzy system is characterized by two pre-defined fuzzy sets: `high` and `low`. Figure 1 shows the membership functions used to define `high` and `low` values for each input. It is worth noticing the log scale used in RTF, which leaves room to handle almost any practical computation time. A very simple linear mapping function is used for the WOT variable since it does not depend on recognition results. It is a parameter defined by the user.




(a) WER (b) log RTF (c) WOT

**Fig. 1**. Membership functions to fuzzify inputs

Figure 2 includes the actual set of fuzzy rules used to define the merit. In the notation used, ↑ means `high` and ↓ is `low`. The combination of fuzzy values in each rule is later to be multiplied by the $z$ value of that rule and later averaged to compute the merit $M$ according to (1). The basic notion for recognition merit stated above (for WER `low` and RTF `low`, merit `high`) is included in rules 1 and 2. Figure 3 graphically shows the merit, as a function of $R(\vec{\Theta})$ and $W(\vec{\Theta})$, computed through the fuzzy logic system described above.

*1. if WER ↓ and RTF ↓ and WOT ↓, then z = 1*
*2. if WER ↓ and RTF ↓ and WOT ↑, then z = 1*
*3. if WER ↓ and RTF ↑ and WOT ↓, then z = 0*
*4. if WER ↓ and RTF ↑ and WOT ↑, then z = 1*
*5. if WER ↑ and RTF ↓ and WOT ↓, then z = 1*
*6. if WER ↑ and RTF ↓ and WOT ↑, then z = 0*
*7. if WER ↑ and RTF ↑ and WOT ↓, then z = 0*
*8. if WER ↑ and RTF ↑ and WOT ↑, then z = 0*

**Fig. 2**. Set of fuzzy rules for merit computation

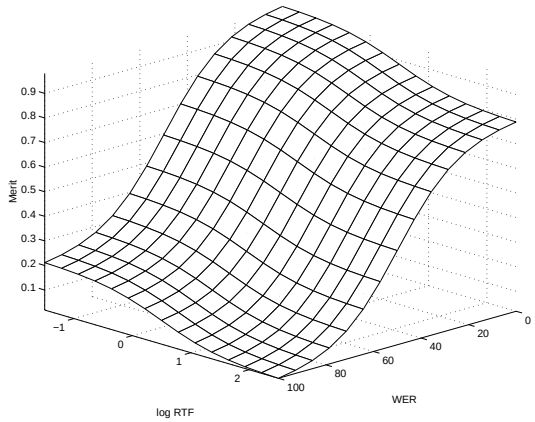



**Fig. 3**. Merit, as a function of WER and RTF, for WOT=0.80

## 3. GRADIENT OPTIMIZATION

In practice, the absolute optimal merit (for WER and RTF = 0) is never reached. However, numeric optimization on real values of $M(\vec{\Theta})$ can help to define a practial optimum. Each element $\theta_i$ of the recognition set-up, a recognition parameter itself, can be optimized in the steepest ascent direction according to:

$$\theta_i(k+1) = \theta_i(k) + \eta_i(k) \cdot \frac{\partial M(\theta_i(k))}{\partial \theta_i} \qquad (2)$$

for $k$ optimization iterations. The partial derivative is computed with respect to each recognition parameter $\theta_1, \theta_2, \ldots, \theta_n$. It should be noticed that the goal here is to maximize the merit and thus the optimization is done in an ascending direction. In practice, the partial derivatives can be approximated by differences. So, for every parameter, the difference is computed by slightly changing the value of the parameter in question and then doing recognition. Afterwards, the new merit is evaluated and it is compared against the initial merit to get the difference:

$$\frac{\partial M(\theta_i)}{\partial \theta_i} \approx \frac{M(\theta_i + \Delta\theta_i) - M(\theta_i)}{\Delta \theta_i}. \qquad (3)$$

The election of the learning rate $\eta_i(k)$ in (2) represents a major implementation question in gradient optimization methods [3]. In our case, optimization is done over a monotonical surface (figure 3). Nevertheless, an adaptative learning rate is needed so optimization can be efficiently conducted. RProp (resilient propagation) [4] is a method that adapts the individual learning rate of each parameter according to the direction of its gradient and not to the actual gradient value. Since this is a method that updates each parameter independently, as if the others did not exist, strictly speaking it is not a gradient search method but a local optimization one. However, in a problem like this, its approximation to the gradient might be sufficient. Since RProp considers the local gradients of previous iterations, it can be thought of as a second order method. RProp defines the individual learning rate of parameter $\theta_i$ at every iteration $k$ according to:

$$\eta_i(k) = \begin{cases} \eta_i(k-1)\cdot\tau^+ & : \quad \frac{\partial M(k-1)}{\partial \theta_i}\cdot\frac{\partial M(k)}{\partial \theta_i} > 0 \\ \eta_i(k-1)\cdot\tau^- & : \quad \frac{\partial M(k-1)}{\partial \theta_i}\cdot\frac{\partial M(k)}{\partial \theta_i} < 0 \\ \eta_i(k-1) & : \quad \text{otherwise} \end{cases} \tag{4}$$

with $\tau^+$ and $\tau^-$ being the incremental and decremental learning weights: $0 < \tau^- < 1 < \tau^+$. The actual recognition parameters are updated according to the direction of the gradient:

$$\Delta\theta_i(k) = \begin{cases} \text{sign}(\frac{\partial M(k)}{\partial \theta_i})\cdot\eta_i(k) & : \quad \frac{\partial M(k-1)}{\partial \theta_i}\cdot\frac{\partial M(k)}{\partial \theta_i} \geq 0 \\ 0 & : \quad \text{otherwise} \end{cases} \tag{5}$$

and search is restarted by setting $\frac{\partial M(k)}{\partial \theta_i} = 0$ when $\frac{\partial M(k-1)}{\partial \theta_i}\cdot\frac{\partial M(k)}{\partial \theta_i} < 0$.

For optimization to start, the set-up $\vec{\Theta}(0)$ is given initial values and a vector of initial updates $\Delta\vec{\Theta}(0)$ is defined. After computing all the local gradients, the values of the set-up $\vec{\Theta}(k)$ are updated all at once according to (5) in a "learning by epoch" fashion. The definition of the initial $\vec{\Theta}(0)$ depends on the particular nature of the recognition system.

## 4. EXPERIMENTAL DEVELOPMENT

The automatic set-up method is tested with two LVCSR applications in two different languages: American English and Japanese. The American English application is regarded as "travel domain". It includes prompted phrases pronounced by American English speakers about requests and answers related to tourist information. The Japanese recognition task here is referred to as "chat". It includes spontaneous conversations held in an informal chatting mode about a number of different topics. Table 1 shows a description of the testing databases.

| | AmEngl | Japanese |
|---|---|---|
| # sentences | 982 | 560 |
| # words | 6152 | 6612 |
| total speech | 0.59 hr | 0.64 hr |

**Table 1**. Testing database configuration

The acoustic model (AM) used for the American English testing is a Gaussian Mixture HMM with 2000 states. After state-tying, there are 24 Gaussian mixtures on average in each state. The front-end parameterization is a 26 LDA-transformed one from an original 38 MFCC. A dictionary with 19k words on its vocabulary is used. The language model (LM) is a tri-gram trained from read and conversational text examples.

For the Japanese case, the AM is also a Gaussian Mixture HMM but with 1000 states. The front-end is also different; it is composed of 25 MFCCs. The Japanese dictionary includes 64k words and the LM is a tri-gram.

Japanese and American English experiments were done using the same recognition engine. The Sony-built recognition engine has no less than 23 recognition parameters on its set-up. Some of them are related to the language weights and insertion penalties that balance the AM and LM scores. Some others are related to the width of the AM beams and LM thresholds used during the search. The sensitivity of the system to each parameter is different but all parameters are important for optimal performance.

To expedite the automatic setting-up, modular and parallel implementation of the method was used. With a modular implementation, the system is more flexible to method changes in the merit evaluation or in the optimization. Three modules are used: merit computation, optimization and experiment scheduler. The scheduler controls the execution of each recognition experiment as well as the optimization and merit evaluations. A parallel implementation allows to distribute the recognition work for merit computation among several different CPUs. For each recognition set-up, accuracy and speed are evaluated separetely. Since recognition accuracy does not depend on computational power, the WER component of the merit is measured regardless of the nature of the CPU or its computational load. The RTF, that depends on the CPU, is measured with a subset of the testing database in a unique and dedicated CPU wich reports a reliable speed comparison between different recognition set-ups. For speed reference, all experimentation here reported was carried out on a Sun Solaris Ultra-Sparc Workstation at 400 MHz.

Since the value of WOT (WER_over_RTF) drastically changes the shape of the optimization function, experiments for different WOT values were done. For 20 epochs, since early stop of the optimization was not used, parameters are automatically updated according to RProp. In this case, the RProp parameters $\tau^+$ and $\tau^-$ are 1.2 and 0.5 respectively. After optimization, there are 20 resulting set-ups, one per epoch, for each WOT value. From those 20 settings, the one that brings highest merit is considered the optimal set-up. Recognition results with the optimal set-ups for different WOT values are shown in table 2. For reference, this table also includes results for set-ups that were adjusted in a "manual" way. Manual adjustment was done by an expert user through extensively testing the settings of each parameter either individually or clustered in small groups with other parameters. Needless to say, manual set-up is a tedious process that requires attentive supervision by the user and its overall result depends heavily on his or her expertise.

## 5. DISCUSSION

According to the results in table 2, our automatic setting-up method can be a good replacement to tedious manual adjustment. As expected, with higher values of WOT more accuracy can be reached. On the other hand, when WOT has a lower value, the resulting set-up allows higher recognition speed. This confirms that WOT is an efficient manner to control the accuracy/speed trade-off. When compared against the manual adjustment, better accuracy and bet-

ter speed can be reached with this method. This is true for both American English and Japanese results.

| **American English Travel Domain** | | | | |
|---|---|---|---|---|
| WER_over_RTF | WER | RTF | epoch | merit |
| 0.80 | 18.99 | 0.90 | 11 | 0.8813 |
| 0.90 | 16.73 | 1.08 | 4 | 0.9061 |
| manual | 17.68 | 0.99 | – | – |
| **Japanese Chat Test** | | | | |
| WER_over_RTF | WER | RTF | epoch | merit |
| 0.70 | 29.46 | 0.86 | 14 | 0.8053 |
| 0.80 | 28.84 | 1.21 | 8 | 0.8068 |
| manual | 29.05 | 0.92 | – | – |

**Table 2**. Results for American English and Japanese testings.

The use of gradient search allows better exploration of the optimization space since it ventures into regions that the expert might have skipped or disregarded during manual adjustment.

Certainly, this method systematizes the parameter optimization process with little human intervention (WOT is still a user-defined parameter). Yet there are several aspects on it open for discussion. The main point subject to debate is regarding the use of the fuzzy merit as cost function. Instead of fuzzy logic, it is feasible to use a purely analytical expression of the merit, possibly as a linear combination of WER and RTF. However, to find the appropriate definition of such expression can be difficult if there is not enough expertise or data to estimate it with. Furthermore, to use an expression as such leaves little room for inclusion of intuition or common sense. In that respect, fuzzy logic is an appealing framework to handle an intuitive measure such as the recognition merit. The fuzzy merit, as it was defined, is versatile enough to handle diverse testing applications. The Japanese and American English tests are certainly very different. Just the fact of using very different languages speaks for itself. In spite of this, no changes to the fuzzy merit evaluation were needed. When designing the fuzzy system for merit computation, general patterns of recognition behavior were followed and room was left for different values of accuracy and speed. This does not guarantee, however, that this setting-up system is general enough to handle any application at any circumstance. Nevertheless, it is clear that general patterns are more likely to be useful than specific implementations when used on diverse applications.

Regarding the optimization method, two points seem arguable: the derivative estimation and RProp. According to (3), to estimate the derivative of a noisy function based on two points seems inaccurate. To use more points in the derivative (linear regression derivative) could be the solution. However this will considerably increase the number of recognition experiments needed for parameter adjustment making optimization very computational expensive, perhaps as resource hungry as the "brute force" approach. Concerning RProp, the values of its $\tau^+$ and $\tau^-$ parameters along with the initial set-up $\vec{\Theta}(0)$ and updates $\Delta\vec{\Theta}(0)$ may change the overall optimization results. Another point worth noticing is that many of the parameters in the set-up are integers while some others are real-valued. Most optimization methods implicitly assume real values in the optimization variables. In our case, optimization needs to be done with rounded values. Strictly speaking, since $\vec{\Theta}$ includes discrete values, the cost function defined in (1) is discrete. This condition makes value-based optimization methods (such as Quickprop [3]) inefficient for this task. R-Prop, since is based on the direction of the gradient not its value, is better suited to handle this problem. Still, to use rounding in the optimized values does not help R-Prop. Further work in developing an optimization method capable of handling integers as well as real values is needed.

As stated above, here we have only considered WER and RTF as our performance attributes. However more attributes can be used in the definition of merit. Memory has already been mentioned but WER can also be decomposed into insertion, substitution or deletion rates. To build a fuzzy merit evaluator considering more figures of performance is straightforward and it could add versatility and precision to the method.

Although this work was carried out using a single recognition engine, none of its components were explicitly considered on the optimization design. It is true that results for a method as this may vary with every recognition engine. However, it is clear that every recognizer has parameters to be adjusted and that in the adjustment some sort of performance trade-off can be found. Even when using the same recognition engine, it cannot be expected that the recognition set-up defined for a certain recognition experiment is going to be of use in another recognition environment. Every time a component of the recognition system, such as AM, LM, dictionary or recognition task, is changed, the engine set-up should be changed accordingly. The use of this method will reduce the need of human supervision in this process but cannot avoid it.

## 6. CONCLUDING REMARKS

The experimentation here reported demonstrates that the method proposed to automatically define the recognition engine set-up conveys good performance, better than manual adjustment, and alleviates the effort in getting the most out of a recognition system for a given application. The adjustment method is based on the definition of a merit function that, for reasons of design and interpretation ease, is computed through a fuzzy logic system. The optimization process is done in a systematic way that, while reducing the extensive experimentation of a "brute force" approach, defines an operating point close to the practical optimum. Further work with this method may include its application to different and more general recognition tasks as well as modifications so that even less user interaction is required.

## 7. REFERENCES


[1] K. Seymore, S. Chen, M. Eskenazi, and R. Rosenfeld, "Language and pronunciation modeling in the CMU 1996 Hub 4 evaluation", in *Proceedings of the 1997 ARPA Speech Recognition Workshop*, 1997.

[2] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial", *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, March 1995.

[3] T. Jervis and W. Fitzgerald, "Optimization schemes for neural networks", Tech. Rep., CUED/F-INFENG/TR 144, Cambridge University Engineering Department, Trumpington Street, Cambridge, England., 1993.

[4] M. Riedmiller and H. Braun, "Rprop- a fast adaptive learning algorithm", Tech. Rep., (Also Proc. of ISCIS VII), University of Karlsruhe, 1992.