Automatic Language Identification using Long Short-Term Memory Recurrent Neural Networks

Javier Gonzalez-Dominguez[1,2], *Ignacio Lopez-Moreno*[1], *Haşim Sak*[1], *Joaquin Gonzalez-Rodriguez*[2], *Pedro J. Moreno*[1]

[1]Google Inc., New York, USA
[2]ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Madrid, Spain
javier.gonzalez@uam.es, jgd@google.com

Abstract

This work explores the use of Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) for automatic language identification (LID). The use of RNNs is motivated by their better ability in modeling sequences with respect to feed forward networks used in previous works. We show that LSTM RNNs can effectively exploit temporal dependencies in acoustic data, learning relevant features for language discrimination purposes. The proposed approach is compared to baseline i-vector and feed forward Deep Neural Network (DNN) systems in the NIST Language Recognition Evaluation 2009 dataset. We show LSTM RNNs achieve better performance than our best DNN system with an order of magnitude fewer parameters. Further, the combination of the different systems leads to significant performance improvements (up to 28%).

1. Introduction

The problem of automatic language identification (LID) can be defined as the process of automatically identifying the language of a given spoken utterance [1]. LID is daily used in several applications such as multilingual translation systems or emergency call routing, where the response time of a fluent native operator might be critical [1] [2].

Even though several high level approaches based on phonotactic and prosody are used as meaningful complementary sources of information [3][4][5], nowadays, many state-of-the-art LID systems still include or rely on acoustic modelling [6][7]. In particular, guided by the advances on speaker verification, the use of i-vector extractors as a front-end followed by diverse classification mechanisms has become the state-of-the-art in acoustic LID systems [8][9].

In [10] we found Deep feeed forward Neural Networks (DNNs) to surpass i-vector based approaches when dealing with very short test utterances ($\leq$3s) and large amount of training material is available ($\geq$20h per language). Unlike previous works on using neural networks for LID [11] [12] [13], this was, to the best of our knowledge, the first time that a DNN scheme was applied at large scale for LID, and benchmarked against alternative state-of-the-art approaches.

Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) [14, 15, 16] have recently been shown to outperform the state of the art DNN systems for acoustic modeling in large vocabulary speech recognition [17, 18]. Recurrent connections and special network units called memory blocks in the recurrent hidden layer in LSTM RNNs make them a more powerful tool to model sequence data than feed forward neural networks and conventional RNNs. The memory blocks contain memory cells with self-connections storing the temporal state of the network which changes with the input to the network at each time step. In addition, they have multiplicative units called gates to control the flow of information into the memory cell and out of the cell to the rest of the network. This allows the network to model temporal sequences such as speech signals and their complex long-range correlations.

In this paper, we propose LSTM RNNs for automatic language identification. Our motivation is that LSTM RNNs' effectiveness in modeling temporal dependencies in the acoustic signal can help learning long-range discriminative features over the input sequence for language identification. To assess the proposed method's performance we experiment on the NIST Language Recognition Evaluation 2009 (LRE'09). We focus on short test utterances (up to 3s). We show that LSTM RNNs perform better than feed forward neural networks with an order of magnitude fewer parameters. Besides, they learn complementary features to DNNs and we get significant improvements by combining the scores from DNN and LSTM RNN systems.

The rest of this paper is organized as follows. Section 2 presents the i-vector based baseline system and the feed forward DNN architecture. Section 3 is devoted to present the LSTM RNN architecture. The fusion and calibration procedure is presented in Section 4. The experimental protocol and datasets used are then described in section 5. Results are discussed in section 6. Finally, conclusions are presented in Section 7.

2. Baseline Systems

To establish a baseline framework, we built a classical i-vector based acoustic system and three different DNNs based LID systems by varying the number of hidden layers. Baseline systems are summarized below and we refer to [10] for a more detailed description.

2.1. i-vector Based LID Systems

The i-vector system follows the standard procedure described in [8]. It is based on an Universal Background Model consisting of 1024 Gaussian components, trained on 39 dimensional PLP coefficients (13 + Δ + $\Delta\Delta$). From Baum-Welch statistics computed over this UBM, we derived a Total Variability (TV) space of 400 dimensions. Our TV model is trained using a PCA followed by 10 EM iterations.

We adopted a classical classification scheme based on Linear Discriminant Analysis followed by Cosine Distance (LDA_CS). Thus, the similarity measure (score) for a given test utterance i-vector w, and the mean i-vector w_L of the language

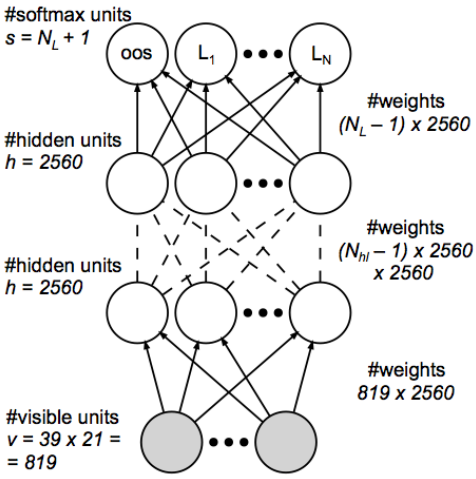

Figure 1: DNN network topology

L is given by

$$S_{w,w_L} = \frac{(A^t w)(A^t w_L)}{\sqrt{(A^t w)(A^t w)}\sqrt{(A^t w_L)(A^t w_L)}} \quad (1)$$

being A is the LDA matrix.

In [10] we provided a more detailed comparison between state-of-the-art i-vector and DNN -based LID system over the Google 5M dataset. In this work we opted for a LDA_CS baseline as it is a widely used technique and offers comparable results with the DNN model on the public LRE'09 dataset [10].

The total number of parameters of the i-vector system accounts for the TV and LDA matrices. It is given by $NFD + D(N_L - 1)$, being N, F, D and N_L the number of Gaussians components (1024), the feature dimension (39) the i-vector dimensions (400) and the number of languages (8). In our model, this makes a total of ∼16M of parameters.

2.2. DNN-based LID System

The DNN architecture used in this work is a fully connected feed-forward neural network [19]. The hidden layers contain units with rectified linear activation functions. The output is configured as a softmax layer with a cross-entropy cost function. Each hidden layer contains h (2560) units while the output layer dimension (s) corresponds to the number of target languages (N_L) plus one extra output for the out-of-set (oos) languages.

The DNN works at frame level, using the same features as the baseline systems described above (39 PLP). The input layer is fed with 21 frames formed by stacking the current processed frame and its ±10 left-right context. Therefore, there are 819 (21 × 39) visible units, v. The number of total weights w, considering N_{hl} hidden layers can be then easily computed as $w = vh + (N_{hl} - 1)hh + sh$. Figure 1 represents the complete topology of the network.

Regarding the training parameters, we used asynchronous stochastic gradient descent within the DistBelief framework [20]. We also fixed the learning rate and minibatch size to 1e-03 and 200 samples.

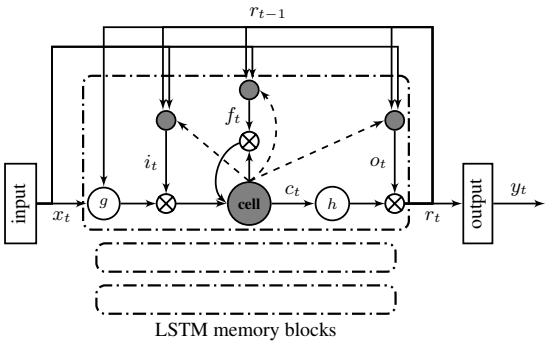

Figure 2: Long Short-Term Memory recurrent neural network architecture. A single memory block is shown for clarity.

Finally, we computed the output scores at utterance level by respectively averaging the log of the softmax output of all its frames (i.e. log of the predicted posterior probabilities).

3. Long Short-Term Memory RNNs

The modern LSTM RNN architecture [14, 15, 16] is shown in Figure 2. The LSTM contains special units called *memory blocks* in the recurrent hidden layer. The memory blocks contain memory cells with self-connections storing the temporal state of the network in addition to special multiplicative units called gates to control the flow of information. The input gate controls the flow of input activations into the memory cell. The output gate controls the output flow of cell activations into the rest of the network. The forget gate scales the internal state of the cell before adding it as input to the cell through self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cell's memory. In addition, the LSTM RNN architecture contains *peephole connections* from its internal cells to the gates in the same cell to learn precise timing of the outputs [16].

With this architecture, LSTM RNNs compute a mapping from an input sequence $x = (x_1, ..., x_T)$ to an output sequence $y = (y_1, ..., y_T)$. They calculate the activations for network units using the following equations iteratively from the time step $t = 1$ to T:

$$i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + W_{ic}c_{t-1} + b_i) \quad (2)$$
$$f_t = \sigma(W_{fx}x_t + W_{fr}r_{t-1} + W_{fc}c_{t-1} + b_f) \quad (3)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{cx}x_t + W_{cr}r_{t-1} + b_c) \quad (4)$$
$$o_t = \sigma(W_{ox}x_t + W_{or}r_{t-1} + W_{oc}c_t + b_o) \quad (5)$$
$$r_t = o_t \odot tanh(c_t) \quad (6)$$
$$y_t = \phi(W_{yr}r_t + b_y) \quad (7)$$

where the W terms denote weight matrices (e.g. W_{ix} is the matrix of weights from the input gate to the input), W_{ic}, W_{fc}, W_{oc} are diagonal weight matrices for peephole connections, the b terms denote bias vectors (b_i is the input gate bias vector), σ is the logistic sigmoid function, and i, f, o and c are respectively the input gate, forget gate, output gate and cell activation vectors, all of which are the same size as the cell output activation vector r, $\odot$ is the element-wise product of the vectors, $tanh$ is the hyperbolic tangent activation function for cell inputs and cell outputs, and ϕ is the softmax output activation function for the LSTM RNN models used in this paper.

The LSTM RNN architecture that we used in this paper contains 512 memory cells. Different than DNNs, the input to the network is just 39-dimensional PLP features calculated at a given time step with no stacking of acoustic frames. The total number of parameters N ignoring the biases can be calculated as $N = n_i \times n_c \times 4 + n_c \times n_c \times 4 + n_c \times n_o + n_c \times 3$, where n_c is the number of memory cells, n_i is the number of input units, and n_o is the number of output units.

We trained the LSTM RNN model using asynchronous stochastic gradient descent (ASGD) and the truncated backpropagation through time (BPTT) learning algorithm [21] within a distributed training framework [18, 22]. Activations are forward propagated for a fixed step time of 20 over a subsequence of an input utterance, the cross entropy gradients are computed for this subsequence and backpropagated to its start. For better randomization of gradients in ASGD and stability of training, we split the training utterances into random chunks of length between 2.5 and 3 seconds. We also set the same target language id sparsely for a chunk, 1 in every 5 frames for the experiments in this paper. The errors are only calculated for the frames that we set a target language id. We used 100 machines for distributed training and in each machine 4 concurrent threads each processing a batch of 4 subsequences. We used an exponentially decaying learning rate of 1e-04. For scoring, we computed an utterance level score for each target language by averaging the log of the softmax outputs for that target language of all the frames in an utterance.

4. Fusion and Calibration

We used multiclass logistic regression in order to combine and calibrate the output of individual LID systems [23]. Let $s_{kL}(x_t)$ be the log-likelihood score for the recognizer k and language L for utterance x_t. We derive combined scores as

$$\hat{s_L}(x_t) = \sum_{k=1}^{K} \alpha_k s_{kL}(x_t) + \beta_L \tag{8}$$

Note that this is just a generic version of the product rule combination, parametrized by α and β. Defining a multiclass logistic regression model for the class posterior as

$$P(L|\hat{s_L}(x_t)) = \frac{exp(\hat{s_L}(x_t))}{\sum_l exp(\hat{s_l}(x_t))} \tag{9}$$

we found α and β to maximize the global log-posterior in a held-out dataset

$$Q(\alpha_1, ..., \alpha_K, \beta_1, ...\beta_N) = \sum_{t=1}^{T} \sum_{l=1}^{N_L} \delta_{tl} P(L|\hat{s_l}(x_t)) \tag{10}$$

being

$$\delta_{tL} \begin{cases} w_L, & \text{if } x_t \in L \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

where w_l ($l = 1, ..., N_L$) is a weight vector which normalizes the number of samples for every language in the development set (typically, $w_L = 1$ if an equal number of samples per language is used). This fusion and calibration procedure was conducted through the FoCal toolkit [24].

5. Datasets and Evaluation Metrics

5.1. The NIST Language Recognition Evaluation dataset

The LRE evaluation in 2009 included data coming from two different audio sources. Besides Conversational Telephone Speech (CTS), used in the previous evaluations, telephone speech from broadcast news was used for both training and test purposes. Broadcast data were obtained via an automatic acquisition system from "Voice of America" news (VOA) where telephone and non-telephone speech is mixed. Up to 2TB of 8KHz raw data containing radio broadcast speech, with the corresponding language and audio source labels were distributed to participants; and a total of 40 languages (23 target and 17 out of set) were included.

Due to the large disparity on training material for every language (from $\sim$10 to $\sim$950 hours) and also, for the sake of clarity, we selected 8 representative languages for which up to 200 hours of audio are available: US English (en), Spanish (es), Dari (fa), French (fr), Pashto (ps), Russian (ru), Urdu (ur), Chinese Mandarin (zh). Further, to avoid misleading result interpretation due to the unbalanced mix of CTS and VOA, all the data considered in this dataset belongs to VOA.

For evaluation, we used a subset of the official NIST LRE 2009 3s condition evaluation set (as for training, we also discarded CTS test segments), yielding a total of 2916 test segments of the 8 selected languages. That makes a total of 23328 trials.

5.2. Evaluation metrics

In order to assess the performance, two different metrics were used. As the main error measure to evaluate the capabilities of one-vs.-all language detection, we use C_{avg} (average cost) as defined in the LRE 2009 [25][26] evaluation plan. C_{avg} is a measure of the cost of taking bad decisions, and therefore it considers not only discrimination, but also the ability of setting optimal thresholds (i. e. calibration). Further, the well-known metric Equal Error Rate (EER) is used to show the performance, when considering only scores of each individual language. Detailed information can be found in the LRE'09 evaluation plan [25].

6. Experimental Results

6.1. Standalone systems performance

In [10] we found DNNs to outperform several different i-vector based systems when dealing with short test durations and large amount of training data (>20h per language). We followed up those experiments by first exploring the use of LSTM RNNs as a natural approach to exploit useful temporal information for LID; and second exploring the effect of varying the number of hidden layers in the DNN architecture.

Table 1 summarizes the results obtained in terms of EER (per language and on average) and C_{avg}. At a first glance, we highlight two major results. First, the proposed LSTM RNN architecture better performance than our best DNN system with 4 hidden layers. This fact is particularly interesting taking into account that the proposed LSTM RNN contains 20 times fewer parameters (see *Size* column in Table 1). Additionally, note from the C_{avg} values that the scores produced by the LSTM RNN model are better calibrated than those produced by DNN or i-vector systems. Second, both neural networks approaches (DNNs and LSTM RNN) surpass the i-vector system performance by $\sim$47% and $\sim$52% in EER and C_{avg} respec-

		Equal Error Rate (EER in %)									
	Size	en	es	fa	fr	ps	ru	ur	zh	EER_{avg}	C_{avg}
i-vector_LDA_CS	∼16M	17.22	10.92	20.03	15.30	19.98	14.87	18.74	10.09	15.89	0.1968
DNN_2_layers	∼8M	12.66	5.04	19.67	8.60	17.84	8.75	14.78	5.54	11.61	0.1727
DNN_4_layers	∼21M	8.53	3.58	16.19	5.82	15.42	6.38	11.24	3.16	8.79	0.1292
DNN_8_layers	∼48M	8.65	3.74	17.22	7.53	16.01	5.59	13.10	4.82	9.58	0.1376
LSTM RNN	∼1M	5.81	3.23	17.46	6.42	12.52	6.16	9.91	5.30	8.35	0.0944
Fusion1	∼22M	5.19	2.16	13.67	4.12	10.82	3.98	8.20	3.91	6.51	0.0693
Fusion2	∼38M	5.34	2.09	12.80	4.24	9.83	4.39	7.62	3.76	**6.26**	0.0654
Fusion3	∼94M	5.42	2.95	12.01	4.40	10.98	4.01	8.20	3.76	6.47	**0.0649**

Table 1: Systems performance per language and average metrics on LRE'09 subset (3s test segments)

tively. This result confirms the ability of the proposed neural networks architectures to exploit discriminative information in large datasets.

A further analysis regarding the optimal *depth* of the DNN system shows that the 4-hidden layer topology outperforms the 2-hidden layers one and more interestingly, the 8-hidden layers topology. In particular, the DNN_4_layers achieved, in average, ∼8% better EER than the DNN_8_layers despite of using just half of the parameters.

6.2. Systems combination performance

In this section we aim to analyze the score correlation among LSTM, DNN and i-vector systems and in particular, how that can lead to a good combination strategy. For this purpose defined three different groups of systems and combined them using the multiclass logistic regression framework presented in Section 4. The three groups defined bellow represent various tradeoffs between performance and number of parameters.

- **Fusion1**: this group is composed by the DNN_4_layers and LSTM RNN systems. This combination strategy allow us to evaluate the fusion capabilities of the proposed DNN and LSTM RNN architectures.
- **Fusion2**: this group incorporates the i-vector system to the compound Fusion1 system. It analyzes the complementarity between neural networks and a classical i-vector approach.
- **Fusion3**: this group includes DNN_2_layers and DNN_8_layers to the systems in Fusion2 to explore a global fusion for all the developed systems.

Fusion results are collected in Table 1. As observed, the combination of the best DNN system and LSTM (Fusion1) gets a >25% gain of performance in terms of C_{avg} with respect to our best individual LSTM RNN system. This fact shows that despite of the presumable similarity of the approaches (both neural nets trained via ASGD), they produce uncorrelated scores that can be successfully combined. Further improvements achieved by Fusion2 highlights the degree of complementary between i-vectors, DNN and LSTM RNN systems. This result is particularly interesting taking into account the gap of performance between Fusion1 and i-vector_LDA_CS. Finally, we present the fusion of all the developed systems in Fusion3. As expected, different DNN topologies exploit correlated information, which turns into a not significant improvement over Fusion2.

7. Conclusions

In this work, we proposed a new approach to Automatic Language Identification (LID) based on Long Short Term Memory (LSTM) Recurrent Neural Networks (RNNs). Motivated by the recent success of Deep Neural Networks (DNNs) for LID, we explored LSTM RNNs as a natural architecture to include temporal contextual information within a neural network system.

We compared the proposed system with an i-vector based system and different configurations of feed forward DNNs. Results on NIST LRE 2009 (8 languages selected and 3s condition) show that LSTM RNN architecture achieves better performance than our best 4 layers DNN system using 20 times fewer parameters (∼1M vs ∼21M). In addition, we found LSTM RNN scores to be better calibrated than those produced by the i-vector or the DNN systems.

This work also shows that both LSTM RNN and DNN systems remarkably surpass the performance of the individual i-vector system. Furthermore, both neural network approaches can be combined leading to a improvement of >25% in terms of C_{avg} with respect to our best individual LSTM RNN system. Our best combined system also incorporates the scores from the i-vector system leading to a total improvement of 28%.

8. References

[1] Y. Muthusamy, E. Barnard, and R. Cole, "Reviewing automatic language identification," *Signal Processing Magazine, IEEE*, vol. 11, no. 4, pp. 33–41, 1994.

[2] E. Ambikairajah, H. Li, L. Wang, B. Yin, and V. Sethu, "Language identification: A tutorial," *Circuits and Systems Magazine, IEEE*, vol. 11, no. 2, pp. 82–108, 2011.

[3] M. Zissman, "Comparison of Four Approaches to Automatic Language Identification of Telephone Speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 4, no. 1, pp. 31–44, 1996.

[4] L. Ferrer, N. Scheffer, and E. Shriberg, "A Comparison of Approaches for Modeling Prosodic Features in Speaker Recognition," in *International Conference on Acoustics, Speech, and Signal Processing*, 2010, pp. 4414–4417.

[5] D. Martinez, E. Lleida, A. Ortega, and A. Miguel, "Prosodic features and formant modeling for an ivector-based language recognition system," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 6847–6851.

[6] P. Torres-Carrasquillo, E. Singer, T. Gleason, A. McCree, D. Reynolds, F. Richardson, and D. Sturim, "The

MITLL NIST LRE 2009 Language Recognition System," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 2010, pp. 4994–4997.

[7] J. Gonzalez-Dominguez, I. Lopez-Moreno, J. Franco-Pedroso, D. Ramos, D. Toledano, and J. Gonzalez-Rodriguez, "Multilevel and Session Variability Compensated Language Recognition: ATVS-UAM Systems at NIST LRE 2009," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 6, pp. 1084–1093, 2010.

[8] N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and R. Dehak, "Language Recognition via i-vectors and Dimensionality Reduction." in *INTERSPEECH*. ISCA, 2011, pp. 857–860.

[9] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matejka, "Language Recognition in iVectors Space." in *INTERSPEECH*. ISCA, 2011, pp. 861–864.

[10] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic Language Identification using Deep Neural Networks," *Acoustics, Speech, and Signal Processing, IEEE International Conference on, to appear.*, 2014.

[11] R. Cole, J. Inouye, Y. Muthusamy, and M. Gopalakrishnan, "Language identification with neural networks: a feasibility study," in *Communications, Computers and Signal Processing, 1989. Conference Proceeding., IEEE Pacific Rim Conference on*, 1989, pp. 525–529.

[12] M. Leena, K. Srinivasa Rao, and B. Yegnanarayana, "Neural network classifiers for language identification using phonotactic and prosodic features," in *Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on*, 2005, pp. 404–408.

[13] G. Montavon, "Deep learning for spoken language identification," in *NIPS workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[15] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[16] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, pp. 115–143, Mar. 2003.

[17] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[18] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," *ArXiv e-prints*, Feb. 2014.

[19] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of Pretrained Deep Neural Networks to Large Vocabulary speech recognition," in *Proceedings of Interspeech 2012*, 2012.

[20] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large Scale Distributed Deep Networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1232–1240.

[21] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for online training of recurrent network trajectories," *Neural Computation*, vol. 2, pp. 490–501, 1990.

[22] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *submitted to INTERSPEECH 2014*, 2014.

[23] N. Brümmer and D. van Leeuwen, "On Calibration of Language Recognition Scores," in *Proc. of Odyssey*, San Juan, Puerto Rico, 2006.

[24] N. Brümmer. Fusion and calibration toolkit [software package]. [Online]. Available: , http://sites.google.com/site/nikobrummer/focal.

[25] NIST, "The 2009 NIST SLR Evaluation Plan," www.itl.nist.gov/iad/mig/tests/lre/2009/LRE09_EvalPlan_v6.pdf, 2009.

[26] N. Brümmer, "Measuring, Refining and Calibrating Speaker and Language Information Extracted from Speech," Ph.D. dissertation, Department of Electrical and Electronic Engineering, University of Stellenbosch,, 2010.