INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Spoken Language Identification using LSTM-based Angular Proximity

G. Gelly, J.L. Gauvain

LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91405 Orsay, France
gelly@limsi.fr, gauvain@limsi.fr

Abstract

This paper describes the design of an acoustic language identification (LID) system based on LSTMs that directly maps a sequence of acoustic features to a vector in a vector space where the angular proximity corresponds to a measure of language/dialect similarity. A specific architecture for the LSTM-based language vector extractor is introduced along with the angular proximity loss function to train it. This new LSTM-based LID system is quicker to train than a standard RNN topology using stacked layers trained with the cross-entropy loss function and obtains significantly lower language error rates. Experiments compare this approach to our previous developments on the subject, as well as to two widely used LID techniques: a phonotactic system using DNN acoustic models and an i-vector system. Results are reported on two different data sets: the 14 languages of NIST LRE07 and the 20 closely related languages and dialects of NIST LRE15. In addition to reporting the NIST Cavg metric which served as the primary metric for the LRE07 and LRE15 evaluations, the average LER is provided.
Index Terms: language identification, LSTM, angular loss

1. Introduction

Spoken language identification (LID) is the task of automatically recognizing the language present in a given speech segment. The LID task can be carried out using either models based on different levels of explicit language description (phones, prosody and words), or models based only on acoustic features without any explicit language description [1].

Phonotatic LID systems have been developed since the early 1990s, when the use of phone-based acoustic likelihoods was proposed for language identification [2, 3]. The basic approach was extended to use parallel phone recognizers with phonotactic characteristics [4], lexical information [5, 6] and phone lattices [7, 8]. Variant approaches based on phone decoding with phonotactic models have been explored for many years and have been shown to provide state-of-the-art results [9, 10, 11].

Many acoustic approaches are based on Gaussian Mixture Models (GMM) among which the i-vector framework [12], originally proposed for speaker recognition [13, 14], has been shown to be one of the most effective methods for LID purposes [15].

The number of spoken languages around the world is estimated to be slightly less than 7000, the LID task can then be quite challenging when it comes to discriminating between closely related languages or dialects of the same language. This was one of the challenges of the last language recognition evaluation organized by the National Institute of Science and Technology (NIST) in 2015 (LRE15 [16]). In preparation for this evaluation, we tested acoustic methods that when combined with the phonotactic system improved language recognition performance [17].

With its innate ability to exploit long range dependencies, Long Short Term Memory (LSTM) neural networks were natural candidates as purely acoustic classifiers. This choice was also motivated by our previous experience with LSTM on Speech Activity Detection [18] and some earlier work on this topic [19] which showed good results on short segments for a limited number of languages. This paper describes our work in designing an acoustic LID system based on LSTM recurrent neural networks. Both phonotatic and i-vector systems serve as baselines for reporting results.

Our first attempt to train multi-class Recurrent Neural Network (RNN) based on LSTM cells gave results that were not competitive for language recognition. To overcome this, we designed a four-step training process called *Divide-and-Conquer* (D&C) training. In [20], we compared the D&C training to a straightforward RNN training and showed that it improved the system performance. In this paper, we introduce an LSTM-based system that directly maps a sequence of acoustic features to a vector in a vector space where angular proximity corresponds to a measure of language/dialect similarity. This new architecture, along with an angular proximity loss function, significantly reduces training time and yields better performance. The performance of this new LSTM-based LID system is compared to the D&C approach and to the phonotatic and i-vector baselines.

The next section describes the language recognizer using the LSTM-based language vector (LV) along with the proposed angular proximity loss function. Section 3 presents and discusses the results obtained on two NIST evaluation data sets. Conclusions are given in Section 4.

2. LSTM-based language classifier

Over the last few years, RNNs and in particular RNNs based on LSTM have been successfully applied to a wide range of classification tasks for which the discriminative information is embedded in a sequence. For the LID task it has been shown in [19] that LSTM-RNN can outperform other LID methods on short utterances for a set of 8 languages. More recently LSTM-RNN have been successfully applied to discriminate related languages and dialects [20]. The LTSM-based classifier proposed here is an extension of this work.

2.1. Coordinated-gate LSTM

Long Short-Term Memory cells were introduced to overcome some of the shortcomings of classical RNNs [21] and were popularized after Graves demonstrated their good performance for optical character recognition and speech sequence labeling [22, 23]. In [18], an improved version of the LSTM cell was proposed in which direct links are added between the three gates of a cell. With these new links the three gates of a cell can interact more efficiently and improve the cell behavior. We call this new model CG-LSTM for Coordinated-Gate LSTM.

http://dx.doi.org/10.21437/Interspeech.2017-1334

As shown in Section 3 this added flexibility allows CG-LSTM cells to outperform standard LSTM cells.

2.2. Standard architecture

In [20] we used a standard architecture (see left diagram in Figure 1) with 2 stacked CG-LSTM layers and a MLP composed of one hidden layer and a softmax output layer. This architecture takes a feature sequence as input and yields a sequence of posterior probabilities estimates for each language as output. Then, to obtain a single classification vector, the geometric mean of all the output vectors in the output sequences is computed.

The D&C training proposed in [20] is used as it was demonstrated to be superior to the classical training method for the LID task.

2.3. Language vector extractor

In order to speed up the RNN training process without reducing the network accuracy we have re-designed the network architecture. Our intuition was that the number of stacked layers (RNN layers + MLP) diluted the gradients during backpropagation and thus slowed the convergence especially during the first training steps. The D&C training process tackles this part of the problem by separating the multi-class problem into n smaller problems that we solved using smaller RNNs that are much easier to train. From this smaller RNNs we obtained a smart initialization point for the complete multi-class RNN training which led to a better and faster training.

Here we introduce an RNN architecture that deals directly with this problem: a language vector (LV) extractor which is depicted on the right diagram in Figure 1. First, the MLP was removed to reduce the overall number of layers in the network and at the same time to obtain more homogeneous gradients since the link between the errors and the LSTM cells is more direct. In the same spirit and to ensure a faster convergence, the outputs of both RNN layers are concatenated. Each layer has a weighting coefficient (α_1 and α_2) as shown on the right diagram in Figure 1). Finally, the vectors of the output sequence are averaged over time, giving a unique output vector for the whole sequence, which is then L2-normalized to lay on the unit hypersphere. As a result, the information only resides in the direction of the vector.

The magnitude coefficients applied to the output of the different layers before the L2-normalization allow the optimization process to easily focus on the most discriminative layer(s) for the classification task at hand, without directly constraining the outputs of the recurrent layers.

The dimension of the final unit vector is equal to the sum of all the layers' dimensions (here $2 \times 124 = 248$).

2.4. Angular proximity loss function (AP)

Inspired by the success of *triplet-loss* introduced in [24] for clustering tasks and for speaker change detection [25], a loss function was designed that can simultaneously train the language vector extractor described above, and a unit vector that serves as the reference direction $\boldsymbol{c}_l$ in the same vector space for each language l.

Then, when given an output vector $\boldsymbol{z}$ for a given input feature sequence, the angular offset θ_l between $\boldsymbol{z}$ and every one of the reference directions $\boldsymbol{c}_l$ is determined:

$$\theta_l(\boldsymbol{z}) = \arccos(\boldsymbol{c}_l \cdot \boldsymbol{z}) \tag{1}$$

The language hypothesis with the highest probability is then given by:

$$l^*(\boldsymbol{z}) = \underset{l \in [1,N]}{\operatorname{argmin}}\, \theta_l(\boldsymbol{z}) \tag{2}$$

where N is the number of language to be identified.

And we define the angular proximity loss $\mathcal{L}$ for a vector $\boldsymbol{z}$ belonging to language l as follows:

$$\mathcal{L}(\boldsymbol{z}, l) = \sum_{l' \in [1,N]}^{l' \neq l} \sigma(\theta_l(\boldsymbol{z}) - \theta_{l'}(\boldsymbol{z})) \tag{3}$$

where $\sigma(x) = 1/(1+\mathrm{e}^{-x})$ is the logistic function which brings faster and better convergence by focusing the training efforts on the cases that are close to the boundaries between languages.

This loss function can be differentiated and we have:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}} = \sum_{l' \in [1,N]}^{l' \neq l} \Delta_{ll'}(\boldsymbol{z}) \left(\frac{\boldsymbol{c}_{l'}}{\sqrt{1 - \delta_{l'}(\boldsymbol{z})^2}} - \frac{\boldsymbol{c}_l}{\sqrt{1 - \delta_l(\boldsymbol{z})^2}} \right) \tag{4}$$

where $\delta_{l'}(\boldsymbol{z}) = \boldsymbol{c}_{l'} \cdot \boldsymbol{z}$ is the dot product between $\boldsymbol{z}$ and $\boldsymbol{c}_{l'}$ and with

$$\Delta_{ll'}(\boldsymbol{z}) = \sigma(\theta_l(\boldsymbol{z}) - \theta_{l'}(\boldsymbol{z}))(1 - \sigma(\theta_l(\boldsymbol{z}) - \theta_{l'}(\boldsymbol{z}))) \tag{5}$$

which corresponds to the contribution of the logistic function in the partial derivatives.

We also have for $l' \in [1, N]$ and $l' \neq l$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}_{l'}} = \frac{\Delta_{ll'}(\boldsymbol{z})}{\sqrt{1 - \delta_{l'}(\boldsymbol{z})^2}} \boldsymbol{z} \tag{6}$$

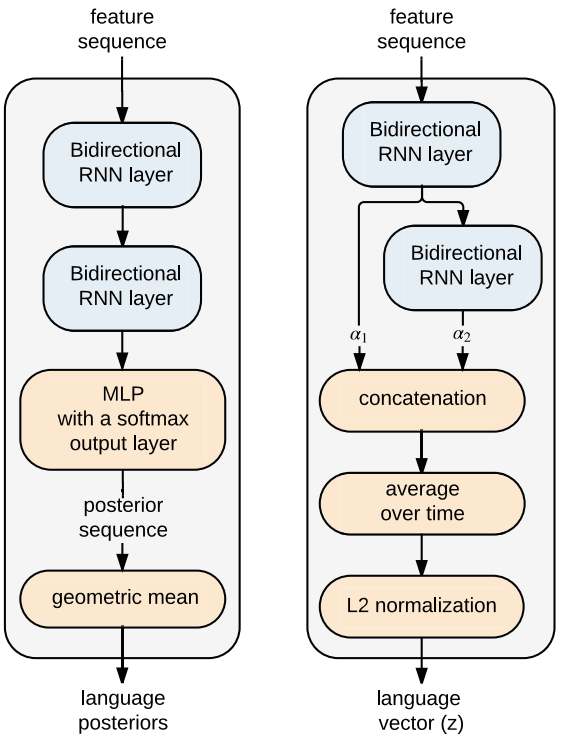

Figure 1: *Left: a standard RNN topology with two stacked layers to estimate posterior probabilities from feature sequences. Right: an RNN topology with two layers to extract, from a feature sequence, a fixed size unit vector based on the output of all the recurrent layers in the network and whose direction is linked to the spoken language.*

and finally

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}_l} = -\sum_{l' \in [1,N]}^{l' \neq l} \frac{\Delta_{ll'}(\boldsymbol{z})}{\sqrt{1-\delta_l(\boldsymbol{z})^2}} \boldsymbol{z} \tag{7}$$

Those last two partial derivatives allow us to use the same gradient descent algorithm to optimize the reference directions for all languages at the same time as we optimize the parameters of the RNN producing the language vectors.

2.5. Training settings

For all the networks trained, the input to the system are 8 PLP coefficients and their first and second derivatives resulting in a 24 dimensional feature vector, with a 10 ms frame rate. VTLN is applied prior to cepstral mean and variance normalization. During system development, it was found beneficial to chop the feature sequence into overlapping segments of 320 frames (i.e. 3.2s) with a shift of 80 frames (i.e. 0.8s) and an overlap of 75% rather than processing the sequence as a whole. This was true independent of the sequence duration. The impact of these two parameters on the LID error rate is shown in Section 3.1.

Training of the various neural network was performed using back-propagation through time ([26], [22]) and as proposed in [27], the SMORMS3 mini-batch gradient descent algorithm was used.

For each training iteration, a small number of training speech segments (about 1000) are randomly selected with an equal number of segments per language to mitigate the effect of the distribution of training segments across languages.

During training, we also keep track of the 200 speech segments that lead to the biggest error rates (with an equal number of worst cases per language) and add them to our mini-batch at each training step.

3. Results

This section presents results on the closed-set tasks of NIST LRE07 and LRE15. Two evaluation metrics are used: the NIST Cavg metric which served as the primary metric for both evaluations ([16], [28]), as well as the average language error rate (LER).

The goal of the NIST LRE07 closed-set task was to identify the spoken language among 14 target languages: Arabic, Bengali, Chinese, English, Farsi, German, Hindustani, Japanese, Korean, Russian, Spanish, Thai, Tamil, and Vietnamese. The evaluation data set is composed of 2158 audio files for each of the 3 speech test durations: 3s, 10s and 30s.

For the LRE15 evaluation, data were grouped into six language clusters (Arabic, Chinese, English, French, Iberian and Slavic) which contain a total of 20 closely related languages or variants of the same language. As detailed in [17], there was a large mismatch between the official evaluation and training data sets which led to poor results on some of the dialects and made it difficult to analyze and compare the performance of the different LID systems. In order to reduce the mismatch, 10% of the files of the evaluation data set were randomly selected and added to the training data. All the LID systems were retrained and tested on the remaining 90% of the evaluation data set.

3.1. Binary classification

A series of preliminary experiments were carried out to find the best neural model and the best way to use it as a LID system. To measure the discriminative power of several neural models, models were trained to separate the Arabic from the other languages in the LRE07 dataset. The models range from a simple MLP to a CG-LSTM based language vector extractor as described in Section 2.3. To have a fair comparison, all the tested models have 50k parameters. The models based on the standard architecture were trained using the cross-entropy loss function (CE) whereas the language vector extractors were trained with the angular proximity loss function (AP) described in Section 2.4.

Table 1: *Comparison of several neural models on a binary classification task (Arabic or not). Error rates on the LRE07 evaluation datasets for the 3 speech test durations. The networks were trained using the cross-entropy loss function (CE) or the angular proximity loss function (AP).*

NN model	*training*		*2-class LER*		
	CE	AP	3 s	10 s	30 s
MLP	✓		33.7	25.0	20.0
RNN	✓		33.9	21.4	12.6
LSTM	✓		27.5	14.4	8.4
CG-LSTM	✓		26.2	13.4	7.5
LSTM		✓	25.9	12.3	7.2
CG-LSTM		✓	24.1	11.2	5.0

Table 1 details the results obtained on the LRE07 evaluation dataset. One can clearly see that the LSTM-based models are much better than a simple MLP or even a standard RNN. One can also see that independently of the loss function and the network architecture, the CG-LSTM model always improves the performance over the standard LSTM cell. In addition, the language vector extractor architecture yields a very significant gain since it yields up to a 33% relative gain on the error rate for the 30 s segments.

Table 2: *Average error rates on the LRE07 evaluation datasets for the binary classification task (Arabic or not) varying the size and shift of overlapping speech segments.*

Overlap ratio (%)	*Segment size*				
	1 s	3.2 s	5 s	10 s	inf
0	16.7	15.4	18.3	19.2	19.8
25	16.7	15.1	17.4	18.8	-
50	17.1	14.6	16.2	17.9	-
75	17.2	14.4	15.0	17.9	-
90	17.0	14.2	14.6	17.5	-

The same task was used to measure the impact of the size of the segments fed to the LID system and the overlap ratio. Table 2 shows the results in terms of overall error rate on the LRE07 evaluation dataset. The advantage of using overlapping chunked segments is easily seen as all setups give better results than using the original full sequence (inf). The chunk duration of 3.2s yields the best results for all overlaps tested. An overlap ratio of 75% was retained for further experiments.

3.2. LID results

This section reports results on the full LID tasks. The performances of the different CG-LSTM based LID systems are compared to two baseline systems: an i-vector system (IV) as im-

plemented in the Kaldi toolkit [29] and a phonotactic system (PH). Detailed descriptions of these two classifiers can be found in [17].

Table 3 reports the results obtained on the LRE07 evaluation data set and on the LRE15 post-evaluation data set. The three RNNs are all based on CG-LSTM cells and are identical in size (about 400k weights[1]).

Table 3: *Results on LRE07 and LRE15 evaluation data with three different CG-LSTM trainings/architectures (standard, D&C, LV) and the baseline systems (phonotactic and i-vector). The best system combinations are also shown.*

System	*LRE07*		*LRE15*	
	LER	*CAVG*	*LER*	*CAVG*
PH	16.2	8.7	23.5	15.1
IV	23.6	12.7	26.6	17.4
CG-LSTM Std	25.6	13.8	30.9	20.8
CG-LSTM D&C	21.9	11.8	22.8	14.6
CG-LSTM LV	20.2	10.9	20.7	13.3
PH + IV	14.1	7.6	18.6	11.6
PH + LV	12.3	6.6	15.9	9.9

One can see that the language vector extractor based on CG-LSTM and trained with the angular proximity loss function improves the LID results and leads to a RNN that performs better than both the i-vector system and the RNN system trained with the D&C method proposed in [20]. One can also see that both acoustic LID systems perform less well than the phonotactic system on LRE07 but the language vector approach leads to a better performance than the phonotactic system on LRE15.

For both evaluation data sets, combining the outputs of the language vector extractor system with the output of the phonotactic system always significantly improves the results. This is also better than the combination of the i-vector and phonotactic systems. System combination is done by taking the geometric mean of the posterior probabilities of the combined systems.

3.3. Training convergence

Figure 2 shows the evolution of the LER of the different CG-LSTM based LID systems during the training phase on the LRE07 train set. As one can see the language error rate drops much faster when using a language vector extractor architecture trained with the angular proximity loss function than with a standard architecture trained with the cross-entropy loss function independently of the use of the D&C strategy.

Finally, with the language vector extractor architecture trained and the angular proximity loss function we were able to train a better CG-LSTM based LID system than with the D&C training strategy in only half the training time. When compared to the standard training, the reduction in training time with the new architecture and training scheme is 66%.

4. Conclusions

This paper introduced a language vector extractor based on a modified LSTM cell along with an angular proximity loss function. This new RNN architecture and training method was evaluated on both the NIST LRE07 and NIST LRE15 data sets. In both cases the proposed scheme significantly outperformed the classical training method for multi-class RNNs as well as the Divide & Conquer training method we proposed in [20].

This error reduction comes with an important reduction in the training duration, the language vector extractor training time is half that needed for the Divide & Conquer method, and a third of the time required for standard training.

The resulting RNN LID system was also compared to a phonotactic system and to an i-vector system. It outperforms the i-vector system on both data sets and outperforms the phonotatic system on the more challenging LRE15 data set, while requiring an order of magnitude fewer parameters. In addition the RNN-based language vector extractor combines well with the phonotatic system, leading to the best results by a significant margin compared to any of the individual systems for both data sets.

[1] In comparison, the number of parameters used for the i-vector and the phonotactic systems is more than 10^7.

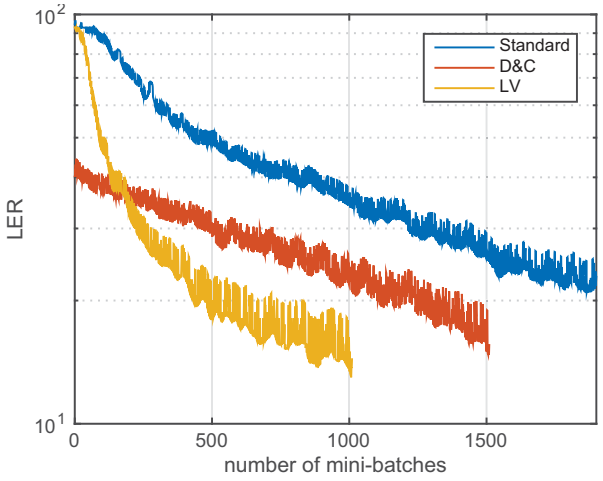

Figure 2: *Evolution of the LER on the LRE07 training dataset for the three different training methods/architectures.*

5. Acknowledgements

We would like to thank Hervé Bredin for all the experiments and the discussions that led us to dig deeper into the language vector extractor approach and the angular proximity loss function that we present here. We also thank Bac Le Viet for training and testing the i-vector system using the Kaldi toolkit.

6. References

[1] H. Li, B. Ma, and K. A. Lee, "Spoken language recognition: from fundamentals to practice," vol. 101, no. 5. IEEE, 2013, pp. 1136–1159.

[2] L. Lamel and J.-L. Gauvain, "Identifying non-linguistic speech features." in *Eurospeech*, 1993.

[3] L. F. Lamel and J.-L. Gauvain, "Language identification using phone-based acoustic likelihoods," in *ICASSP*, 1994.

[4] M. A. Zissman *et al.*, "Comparison of four approaches to automatic language identification of telephone speech," vol. 4, no. 1, 1996, p. 31.

[5] D. Matrouf, M. Adda-Decker, L. Lamel, and J.-L. Gauvain, "Language identification incorporating lexical information." in *ICSLP*, vol. 98, 1998, pp. 181–184.

[6] S. Kadambe and J. Hieronymus, "Language identification with phonological and lexical models," in *Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1995, pp. 3507–3510.

[7] J.-L. Gauvain, A. Messaoudi, and H. Schwenk, "Language recognition using phone latices." in *InterSpeech*, 2004.

[8] D. Zhu and M. Adda-Decker, "Language identification using lattice-based phonotactic and syllabotactic approaches," in *Speaker and Language Recognition Workshop, Odyssey*, 2006, pp. 1–4.

[9] M. F. BenZeghiba, J.-L. Gauvain, and L. Lamel, "Improved n-gram phonotactic models for language recognition." in *InterSpeech*, 2010, pp. 2710–2713.

[10] M. F. BenZeghiba, J. Gauvain, and L. Lamel, "Phonotactic language recognition using MLP features," in *InterSpeech*, 2012, pp. 2041–2044.

[11] M. F. BenZeghiba, J.-L. Gauvain, and L. Lamel, "Fusing language information from diverse data sources for phonotactic language recognition." in *Odyssey*, 2012, pp. 346–352.

[12] D. Najim, K. Patrick, D. Reda, D. Pierre, and O. Pierre, "Front-End Factor Analysis for Speaker Verification," vol. 19, no. 4. IEEE, 2011, pp. 788–798.

[13] P. Kenny, "Bayesian speaker verification with heavy-tailed priors." in *Odyssey*, 2010, p. 14.

[14] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems." in *InterSpeech*, 2011, pp. 249–252.

[15] D. Martınez, O. Plchot, L. Burget, O. Glembek, and P. Matejka, "Language recognition in ivectors space," 2011, pp. 861–864.

[16] NIST, "The 2015 nist language recognition evaluation plan (lre15)," 2015, http://www.itl.nist.gov/iad/mig//tests/lre/.

[17] G. Gelly, J. Gauvain, L. Lamel, A. Laurent, V. Le, and A. Messaoudi, "Language recognition for dialects and closely related languages," 2016.

[18] G. Gelly and J.-L. Gauvain, "Minimum word error training of rnn-based voice activity detection," 2015.

[19] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. J. Moreno, "Automatic language identification using long short-term memory recurrent neural networks." 2014.

[20] G. Gelly, J.-L. Gauvain, V. Le, and A. Messaoudi, "A divide-and-conquer approach for language identification based on recurrent neural networks," 2016, pp. 3231–3235.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, no. 8. MIT Press, 1997, pp. 1735–1780.

[22] A. Graves, *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, vol. 385.

[23] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6645–6649.

[24] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[25] H. Bredin, "TristouNet: Triplet Loss for Speaker Turn Embedding," in *ICASSP 2017, IEEE International Conference on Acoustics, Speech, and Signal Processing*, New Orleans, USA, March 2017.

[26] P. J. Werbos, "Backpropagation through time: what it does and how to do it," vol. 78, no. 10. IEEE, 1990, pp. 1550–1560.

[27] S. Funk, "Rmsprop loses to smorms3," 2015, http://sifter.org/~simon/journal/20150420.html.

[28] NIST, "Nist lre-2007 evaluation plan," 2007, http://www.itl.nist.gov/iad/mig/tests/lre/2007/LRE07EvalPlan-v8b.pdf.

[29] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.