Unsupervised Adaptation of Recurrent Neural Network Language Models

Citation for published version:
Gangireddy, SR, Swietojanski, P, Bell, P & Renals, S 2016, Unsupervised Adaptation of Recurrent Neural Network Language Models. in Interspeech 2016. pp. 2333-2337, Interspeech 2016, San Francisco, United States, 8/09/16. https://doi.org/10.21437/Interspeech.2016-1342

Digital Object Identifier (DOI):
10.21437/Interspeech.2016-1342

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Interspeech 2016

Unsupervised Adaptation of Recurrent Neural Network Language Models

Siva Reddy Gangireddy, Pawel Swietojanski, Peter Bell and Steve Renals

Centre for Speech Technology Research, University of Edinburgh, Edinburgh EH8 9AB, UK

s.gangireddy@sms.ed.ac.uk, {p.swietojanski, peter.bell, s.renals}@ed.ac.uk

Abstract

Recurrent neural network language models (RNNLMs) have been shown to consistently improve Word Error Rates (WERs) of large vocabulary speech recognition systems employing n-gram LMs. In this paper we investigate supervised and unsupervised discriminative adaptation of RNNLMs in a broadcast transcription task to target domains defined by either genre or show. We have explored two approaches based on (1) scaling forward-propagated hidden activations (Learning Hidden Unit Contributions (LHUC) technique) and (2) direct fine-tuning of the parameters of the whole RNNLM. To investigate the effectiveness of the proposed methods we carry out experiments on multi-genre broadcast (MGB) data following the MGB-2015 challenge protocol. We observe small but significant improvements in WER compared to a strong unadapted RNNLM model.

Index Terms: RNNLM, LHUC, unsupervised adaptation, fine-tuning, MGB-Challenge

1. Introduction

Until recently most large vocabulary continuous speech recognition (LVCSR) systems used n-gram language models (LMs) for both first pass decodes and for later n-best list or lattice rescoring with either unpruned or larger LMs, usage of which in the first pass would be otherwise computationally too expensive. However, n-gram LMs, even with standard smoothing and back-off techniques, tend to suffer from data sparsity and lack of generalization ability for unseen sequences of words. In contrast to n-grams, neural network language models (NNLMs) are better able to extrapolate predictions for word sequences unseen at the training stage, and they have been found to be complementary to n-grams [1, 2]. In particular, recurrent neural network language models (RNNLMs) have been shown to consistently improve the perplexity (PPL) and speech recognition word error rates (WER), compared to a standalone usage of n-gram LMs [3, 4, 5, 6, 7, 8, 9, 10].

Broad coverage (or background) LMs are typically trained on large amounts of text data comprising a variety of domains and topics with the intention of making the LM well matched to the unseen testing conditions (in terms of a domain, topic or data source). A matched LM is more likely to bring improvements in the final system accuracy. In many scenarios, however, it is often the case that availability of large amounts of in-domain data for LM training is limited; and to match test conditions a background LM is often first estimated from a large amount of out-of-domain (OOD) text and then interpolated with a smaller in-domain LM. These approaches still rely on identifying a sub-corpus of in-domain material: alternatively, LM adaptation can be carried out explicitly using unsupervised approaches to adapt the language model to the test data at hand.

A number of methods have been proposed to adapt RNNLMs. For example, Chen et al. [11] explored explicit adaptation of RNNLMs to genre and topic using several methods, including fine-tuning on in-domain (genre specific data), the use of a meta-data genre code as an additional input feature, as well as the automatic extraction of topic representations as an additional input feature (computed by either latent Dirichlet allocation [12], probabilistic latent semantic analysis or hierarchical Dirichlet process modelling). The work of Chen et al. was carried out on the multi-genre broadcast (MGB) data used in the MGB challenge [13] and experiments were carried out at both the genre level and show level, with show-level adaptation consistently out-performing genre-level adaptation.

In Multi-Domain RNNLMs [14], a bottleneck (or compression) layer is inserted between the hidden and the output layer, which is then estimated on adaptation data. A domain feature vector is connected to the newly added compression layer, where each dimension in the feature vector represents one domain. A single RNNLM is trained to adapt to multiple domains. In factored RNNLMs, the RNNLMs are provided with some structural information by appending structural feature vectors (POS, Lemma and Stem) to the input feature vectors [15]. In context dependent-RNNLMs [16], context is enhanced by providing the RNNLM with topic proportions computed from fixed number of words preceding the current word. The maximum entropy framework is used to adapt the LMs to the topic and syntactic structure of the sentence [17].

LMs can be also adapted to a target domain using information retrieval methods [18]. In [19], n-grams are adapted to a target domain by merging the counts from ASR transcriptions and language model data. There has been also some work on feed-forward NNLMs domain adaptation by adding an adaptation linear layer between the projection and hidden layers [7].

In this paper, we investigate unsupervised adaptation of RNNLMs to a specific show, performing experiments on the MGB challenge transcription task [13]. The MGB data, which consists of subtitled BBC television broadcasts, is provided with metadata that enables both the show and its genre to be identified. (The genre information is provided by the BBC, according to their standard ontology.) In our experiments we have focused on the adaptation of the RNNLMs to a specific show only, for which we investigate two RNNLM adaptation methods. The first one relies on learning show-dependent amplitudes of the hidden unit contributions (LHUC) [20]. The second approach directly updates the parameters of the background RNNLM. In this work we also discuss the potential difficulties of adapting RNNLMs when updating all parameters using un-

This research was supported by: EPSRC Programme Grant EP/I031022/1, *Natural Speech Technology* (NST); the Core Research for Evolutional Science and Technology (CREST) from the Japan Science and Technology Agency (JST) (*uDialogue* project); and the European Union under H2020 project *SUMMA*, grant agreement 688139. The NST research data collection may be accessed at http://datashare.is.ed.ac.uk/handle/10283/786.

supervised 1-best hypotheses from decoding lattices.

This paper is organised as follows: Section 2 describes the RNNLM architecture and details of training. In Section 3, we briefly describe adaptation methods used in this work. Experimental setup is given in Section 4. Discussions of experimental results is given in Section 5 which are followed by conclusions and future work included in Section 6.

2. Recurrent Neural Network Language Model

The architecture of the RNNLM is shown in Fig 1. The inputs to the network at time t are the index of the previous word, encoded using 1 of N coding and the state of the hidden layer at time t–1. The hidden activations and probability distribution in the output layer are computed as follows:

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1}) \tag{1}$$

$$y_t = g(W_{yh}h_t), \tag{2}$$

Where x_t is the input vector, h_{t-1} is the state of the hidden layer at t–1, h_t is the state of the hidden layer at time t and y_t is the modelled posterior probability distribution. f and g are *sigmoid* and *softmax* functions, respectively. The network is thus parametrised by $\theta = \{W_{hx}, W_{hh}, W_{yh}\}$. The parameters of the network are learned using back propagation through time (BPTT) algorithm [21]. In this work truncated variant of BPTT algorithm is used in which at each time step t the error is propagated fixed number of steps back in time (set to 4 in this work). The parameters of the network are learned by optimizing the cross-entropy between the output and target probability distributions.

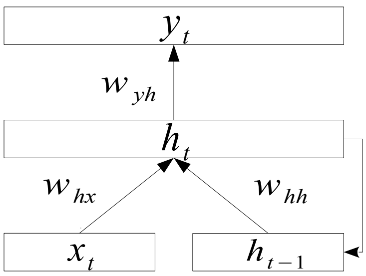

Figure 1: Recurrent neural network language model

3. RNNLM Adaptation

In general, the background LMs are estimated from large amounts of data covering various aspects of broadcast data. We outline two methods to explicitly adapt the RNNLM to a genre, topic, or show: LHUC described in Section 3.1 and fine-tuning the whole model described in Section 3.2.

3.1. Learning Hidden Unit Conributions (LHUC)

In LHUC [20], the hidden activations of the RNNLM are scaled by a vector of adaptation parameters $r_m \in \mathbb{R}^M$ for mth show. The effective range of scaling parameters r_m may be additionally constrained by applying some additional element-wise nonlinearity, $a(r_m)$. After scaling, the output of hidden layer is defined as follows:

$$h_t^{'} = h_t \circ a(r_m) \tag{3}$$

$$a(c) = \frac{2}{1+e^{-c}} \tag{4}$$

Where $h_t^{'}$ are the hidden activations after scaling and $\circ$ denotes element-wise multiplication. The re-parametrisation function in (4) is defined as a sigmoid with amplitude of 2.0, which gives an effective scaling range of $[0, 2]$ [22]. This allows the hidden units to be re-weighted according to their relative importance in modelling the show-specific distribution over sequences of words. The potential advantages of this method lies in the lower number of adaptation parameters θ_m^{lhuc} (for our models 0.00001% of θ) and robustness against potential over-fitting due to learned feature detectors are not updated, which is a desired property when adapting with small amounts of noisy adaptation targets. In this work, we have found it beneficial to update only forward-pass activations for adaptation, which are passed then unscaled to the recurrent layer $h_t^{'}$, in order to avoid modifying learned history.

3.2. Fine-tune

The parameters of an unadapted RNNLM, θ may be fine-tuned on show specific adaptation data (obtained from a 1-best decoding in our work). The set of adapted parameters for a show m are $\theta_m^{fine-tune} = \{W_{hx}^{'}, W_{hh}^{'}, W_{yh}^{'}\}$. These parameters are learned using the standard BPTT algorithm. Since we are fine-tuning the parameters on relatively small amounts of data there is a possibility that the RNNLM over-fit. We experimentally searched for the optimal learning rates on development set, and we report the numbers for both high and low learning rates. The experimental results are given in Section 5.2.

4. Experimental Setup

To investigate the effectiveness of RNNLMs and adapted RNNLMs we rescored 100-best lists obtained in the MGB Challenge transcription task [23, 13]. The details of acoustic and language models are given below.

4.1. Acoustic Models

Acoustic models were trained on 640 hours of MGB challenge multi-genre broadcast data [23, 13], selected from an unfiltered training set of about 1600 hours of audio (collected from 2008). GMMs were trained on filterbank+pitch features following the standard Kaldi recipe [24]. A six-layer DNN with 2048 units in each layer was used to compute the posterior probability of tied states obtained from the GMM acoustic models. Cross-entropy was followed by two iterations of sequence training [25] (we did not regenerate lattices after first iteration, as done in [25]).

4.2. Language Models

The MGB Challenge provided a 640M token corpus of language model training data that contains BBC subtitle data recorded during 1979–2013, all obtained from pre-recorded (rather than live) subtitling. The acoustic training data for the MGB Challenge includes about 10M transcribed words (from both live and prerecorded subtitles): this data was not used to train the baseline language models, but was only used for supervised adaptation experiments. Before training the LMs the text data was normalised, with numbers converted to text-form and abbreviations converted to sequences of letters.

The MGB Challenge development set comprised 47 shows (28 hours of audio), called **dev.full**. In this work the parameters of both n-gram and RNNLM are tuned on **dev.full**. A total of four eval test sets are available for different tasks. In this work we use eval test set for transcription task, referred to as **eval.task1**, which consists of 16 shows (11 hours of audio).

A pruned 3-gram[1] was used in the first pass recognition to generate lattices and n-best lists. A Kneser-Ney smoothed 3-gram LM [26] was trained (using SRILM [27]) on 640M words, with a vocabulary of the 150k most frequent words. This resulted in a perplexity of 175.39 and a WER of 31.0% on **dev.full**.

RNNLMs were also trained on 640M tokens of BBC subtitle text data. Due to computational complexity of training RNNLMs on a vocabulary of 150K words, we trained the RNNLMs by creating input and output short-lists consists of the most frequent words from the 150K vocabulary: in the current work the input and output short-list sizes were 64K and 30K respectively. Both input and output layers had an extra node to compute the probability of out-of-short-list words, represented as <oos>. During PPL computation and n-best list rescoring the probability of <oos> node is distributed equally among all short-list words. The RNNLM is trained with a batch size of 256 and learning rate of 2.0[2]. The parameters of RNNLM were computed by optimizing the cross entropy between output and target probability distributions. The hidden layer consisted of 512 nodes. RNNLMs were trained on GPUs using the Cambridge RNNLM toolkit [28, 29].

For LHUC adaptation, as described in Section 3.1, the scaling parameters were estimated on a first-pass 1-best decoding hypotheses. During adaptation only the LHUC scaling parameters were updated. One set of scaling parameters were estimated for each show in the **dev.full** and **eval.task1** test sets. In addition to adapting the RNNLMs on the 1-best transcripts we also conducted oracle experiments using the reference transcripts. A learning rate of 1.0 (per sample learning rate=3.9×10^{-3}) was used to learn the LHUC parameters. The show-specific adaptation parameters were reused during n-best rescoring.

In the fine-tuning adaptation method, we alter all parameters of RNNLM on the first-pass 1-best decoding. Since we are adapting the RNNLMs on small amounts of adaptation data this method is not robust against over-fitting. We performed a number of experiments by varying the learning during fine-tuning. We report results using learning rates of 0.1 and 1.0.

5. Results and Discussion

The RNNLMs and adapted RNNLMs were applied by rescoring 100-best lists from the **dev.full** and **eval.task1** transcription test sets of the MGB Challenge. The RNNLM scores were interpolated with the 3-gram scores. Since the 3-grams and RNNLMs are trained on same of amount of data the interpolation coefficients were set to 0.5 for all the ASR experimetns. To investigate how the amount of data used to train the RNNLMs affect possible adaptation gains, we trained two RNNLMs on two data sets comprising either full 640M (RNNLM-640M) or limited 40M (RNNLM-40M) tokens. We used an interpolation coefficient of 0.3 for ASR experimetns involving RNNLM-40M LMs.

[1]pruning factor=$1e^{-7}$

[2]Per sample learning rate=7.8125×10^{-3}

5.1. LHUC

The WERs on **dev.full** and **eval.task1** using LHUC adaptation are given in Table 1. In the first row of the Table 1, the WERs using a pruned 3-gram LM are given. After rescoring with the full 3-gram LM we can observe 1.6% and 1.4% absolute improvements on **dev.full** and **eval.task1**, respecively. The WERs of the RNNLM trained on 640M are given in third row of Table 1, 3-gram+RNNLM-640M. With RNNLM-640M we can observe 0.7% absolute improvements on both **dev.full** and **eval.task1**. With an RNNLM trained on 40M tokens we can observe 0.1% and 0.2% absolute improvements compared to the full 3-gram on **dev.full** and **eval.task1**, respectively. LHUC adaptation improves the interpolated 3-gram and RNNLMs by 0.1% absolute, for both RNNLM training cases on **dev.full** and for RNNLM-640M on **eval.task1**. To estimate the bounds on the possible improvements with the proposed LHUC method, we report the WERs resulting from adapting the RNNLMs on reference transcripts of **dev.full** and **eval.task1**. In this scenario (3-gram+RNNLM-640M-lhuc-oracle) we observe 0.2% absolute improvements both on **dev.full** and **eval.task1**. For RNNLM-40M scenario the improvement was 0.1% absolute.

Model	dev.full	eval.task1
3-gram-pruned	32.6	33.6
3-gram-rescored	31.0	32.2
3-gram+RNNLM-640M	30.3	31.5
3-gram+RNNLM-640M-lhuc-1best	30.2	31.4
3-gram+RNNLM-640M-lhuc-oracle	30.1	31.3
3-gram+RNNLM-40M	30.9	32.0
3-gram+RNNLM-40M-lhuc-1best	30.8	32.0
3-gram+RNNLM-40M-lhuc-oracle	30.8	31.9

Table 1: % WERs of RNNLM and adapted RNNLM by LHUC method. The RNNLMs are trained on 640M and 40M tokens. For adaptation, the 1-best decoding and the reference transcripts of **dev.full** and **eval.task1** are used.

5.2. Fine-tuning

Model	dev.full	eval.task1
3-gram-pruned	32.6	33.6
3-gram-rescored	31.0	32.2
3-gram+RNNLM-640M	30.3	31.5
3-gram+RNNLM-640M-finetune-1best	30.2	31.4
3-gram+RNNLM-640M-finetune-oracle	29.9	31.1
3-gram+RNNLM-40M	30.9	32.0
3-gram+RNNLM-40M-finetune-1best	30.8	32.0
3-gram+RNNLM-40M-finetune-oracle	30.7	31.8

Table 2: % WERs of RNNLM and adapted RNNLM by the fine-tuning method. During fine-tuning a learning rate of 0.1 is used. For adaptation, the 1-best decoding and the reference transcripts of **dev.full** and **eval.task1** are used.

In Table 2, we report the WERs on **dev.full** and **eval.task1** by fine-tuning the parameters of RNNLM-640M and RNNLM-40M. The 3-gram and RNNLM baselines are same as above. By fine-tuning the parameters of an RNNLM trained on 640M tokens, we can observe 0.1% absolute gains on both **dev.full** and **eval.task1**. Similar improvements were obtained by fine-tuning the parameters of RNNLM-40M. Similarly to LHUC scenario, we also perform oracle adaptation experiments with fine-tuning method. The numbers for those experiments and RNNLM-640M model are reported in the fifth row of Table 2 which shows 0.3% and 0.4% absolute improvement on **dev.full** and **eval.task1**, respectively. Similarly result (0.2% absolute WER improvement) for RNNLM-40M is reported in the eight of row Table 2.

5.3. Discussion

Tables 1 and 2 show that both LHUC and fine-tuning adaptation methods improve the WER by 0.1% absolute (0.3% relative) for RNNLMs trained on either training scenario (640M vs. 40M tokens). The improvements are small but consistent across test sets. To find the statistical significance of improvements, we performed matched pair sentence segment word error (MPSSWE) [30] tests for the considered adaptation methods and baselines, for the RNNLM-640M case. The statistical significance test reveals the reported improvements, though small, are significant at $p < 0.001$ level. It is due to the fact that both test sets are relatively large – **dev.full** consists of 200K tokens or 28 hours of speech and *eval.task1* consists of 80K tokens or 11 hours of speech.

As discussed in Section 3.1, LHUC is robust against over-fitting, since there are far fewer adaptation parameters than the total number of parameters in the RNN, and because feature receptors are not modified. This is not the case with fine-tuning, in which all the parameters of RNN are altered based on small amounts of adaptation data. It is thus likely that the RNNLM can over-fit the adaptation data when fine-tuning adaptation is used. For the results reported in Table 2, we used a small learning rate of 0.1 (per sample learning rate=3.9×10^{-4}), during adaptation. To investigate the effect of learning rate, we adapted the RNNLM trained on 640M tokens to a target show with a learning rate of 1.0 (per sample learning rate=3.9×10^{-3}). In the first row of the Table 3 we can observe 1.4% absolute improvement on **dev.full**, by adapting the RNNLM on reference transcripts. In the second row of Table 3, we can observe that adaptation on the 1-best decoding with high learning rate has a higher WER than the baseline. In Table 3 we can also observe the PPLs before and after adaptation. After adaptation we can observe 72.3% and 59.4% relative improvements over the baseline on reference and 1-best transcripts, respectively. The improvements on the reference transcripts suggest that, lower WERs in the unsupervised adaptation setting may be obtained once the adaptation process is properly regularised.

Model	dev.full	
	PPL	WER
3-gram+RNNLM-640M-finetune-oracle	153.12/42.27	28.9
3-gram+RNNLM-640M-finetune-1best	202.43/81.99	30.6

Table 3: % WERs baseline RNNLM and adapted RNNLM by fine-tuning method. With a learning rate of 1.0

In Table 1 and Table 2 we report the average WERs of all the shows in **dev.full** and **eval.task1**. Given we adapted the RNNLMs at show level, we also looked at the WER (%) improvements at each show level. After adaptation, both proposed methods improve the WERs of almost all the shows, with fewer than 5 (out of 47) shows with increased WERs after adaptation.

In the MGB Challenge transcription task we also have access to about 10M tokens from the transcriptions of the acoustic training data. Table 4 reports WERs on **dev.full** and **eval.full** obtained by adapting the RNNLMs on this data. Both the 3-gram and RNNLM are adapted using linear interpolation. The full 3-gram LM is interpolated with 3-gram trained on the acoustic training transcripts data, with an interpolation coefficient of 0.9. The RNNLM-640M is interpolated with RNNLM trained on the acoustic training transcripts with an interpolation coefficient of 0.9. From Table 4, we can observe that supervised adaptation improves the baseline by 0.1% absolute. This is a similar improvement to unsupervised adaptation on the test data reported above.

Model	dev.full	eval.task1
3-gram+RNNLM-640M	30.3	31.5
3-gram-adapt+RNNLM-640M-adapt	30.2	31.4

Table 4: % WERs of RNNLM and RNNLM adapted on 10M tokens of acoustic transcripts

6. Conclusions and Future Work

We have investigated unsupervised adaptation of RNNLMs to the test show in multi-genre broadcast transcription task, following the MGB Challenge protocol. We have investigated two adaptation scenarios – LHUC and fine-tuning. Our experimental results indicate that WER reductions arising from unsupervised test-only adaptation using either LHUC or fine-tuning are small but statistically significant.

Our current unsupervised adaptation approach gives equal weight to both correctly recognized and misrecognized words in the 1-best decoding. The influence of errors during adaptation could be reduced by scaling the gradients in proportion to confidence scores of each word. In addition, as discussed above, there is some potential in combining fine-tuning adaptation with larger rates and appropriate regularization (e.g. KL-divergence regularization [31, 32]) or confidence measures. It would also be possible to explore fine-tuning only some parameter subsets [32]. Finally, significant amounts of manually generated metadata are available for broadcast transcription and it should be possible to exploit this information to better aid adaptation process.

7. References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[2] H. Schwenk, “Continuous space language models,” *Computer Speech & Language*, vol. 21, no. 3, pp. 492–518, 2007.

[3] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *Proc IEEE ICASSP*, 2011, pp. 5528–5531.

[4] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, “Recurrent neural network based language model,” in *Proc Interspeech*, 2010, pp. 1045–1048.

[5] P. Bell, H. Yamamoto, P. Swietojanski, Y. Wu, F. McInnes, C. Hori, and S. Renals, “A lecture transcription system

combining neural network acoustic and language models," in *Proc. Interspeech*, Lyon, France, Aug. 2013.

[6] S. Gangireddy, F. McInnes, and S. Renals, "Feed forward pre-training for recurrent neural network language models," in *Proc. Interspeech*, Sep. 2014, pp. 2620–2624.

[7] J. Park, X. Liu, M. J. F. Gales, and P. C. Woodland, "Improved neural network based language modelling and adaptation," in *Proc Interspeech*, 2010, pp. 1041–1044.

[8] X. Chen, X. Liu, M. Gales, and P. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition," pp. 5411–5415, 2015.

[9] X. Liu, Y. Wang, X. Chen, M. Gales, and P. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *in Proc. of ICASSP*, pp. 4908–4912.

[10] S. Gangireddy, S. Renals, Y. Nankaku, and A. Lee, "Prosidically-enhanced recurrent neural network language models," in *Proc. Interspeech*, Sep. 2015.

[11] X. Chen, T. Tan, X. Liu, P. Lanchantin, M. Wan, M. J. F. Gales, and P. C. Woodland, "Recurrent neural network language model adaptation for multi-genre broadcast speech recognition," in *in Proc. of INTERSPEECH*, 2015, pp. 3511–3515.

[12] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[13] P. Bell, M. Gales, T. Hain, J. Kilgour, P. Lanchantin, X. Liu, A. McParland, S. Renals, O. Saz, M. Wester, and P. C. Woodland, "The mgb challenge: Evaluating multi-genre broadcast media recognition," in *Proc. ASRU*, Dec 2015.

[14] O. Tilk and T. Alumäe, "Multi-domain recurrent neural network language model for medical speech recognition," in *Proceedings of the Sixth International Conference Baltic HLT*, 2014, pp. 149–152.

[15] Y. Wu, X. Lu, H. Yamamoto, S. Matsuda, C. Hori, and H. Kashioka, "Factored language model based on recurrent neural network," in *Proceedings of COLING*, 2012, pp. 2835–2850.

[16] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *SLT*. IEEE, 2012, pp. 234–239.

[17] S. Khudanpur and J. Wu, "Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling," *Computer Speech & Language*, vol. 14, no. 4, pp. 355–372, 2000.

[18] L. Chen, J. L. Gauvain, L. Lamel, and G. Adda, "Unsupervised language model adaptation for broadcast news," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 1, 2003, pp. I–220–I–223 vol.1.

[19] M. Bacchiani and B. Roark, "Unsupervised language model adaptation," in *Proceedings. (ICASSP '03)*, vol. 1, April 2003, pp. I–224–I–227 vol.1.

[20] P. Swietojanski, J. Li, and S. Renals, "Learning hidden unit contributions for unsupervised acoustic model adaptation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1–1, 2016.

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

[22] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proc. IEEE Workshop on Spoken Language Technology*, Lake Tahoe, USA, Dec. 2014.

[23] P. Bell and S. Renals, "A system for automatic alignment of broadcast media captions using weighted finite-state transducers," in *Proc. ASRU*, Dec 2015.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit." IEEE Signal Processing Society, 2011.

[25] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Lyon, France, Aug. 2013.

[26] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proc. IEEE ICASSP*, vol. I, 1995, pp. 181–184.

[27] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *Proceedings of INTERSPEECH*, 2002.

[28] X. Chen, X. Liu, Y. Qian, M. J. Gales, P. Woodland *et al.*, "Cued-rnnlm–an open-source toolkit for efficient training and evaluation of recurrent neural network language models," 2016.

[29] X. Chen, Y. Wang, X. Liu, M. J. Gales, and P. C. Woodland, "Efficient gpu-based training of recurrent neural network language models using spliced sentence bunch." in *Proc. of INTERSPEECH*, 2014, pp. 641–645.

[30] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. IEEE ICASSP*, May 1989, pp. 532–535 vol.1.

[31] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *ICASSP*, 2013, pp. 7893–7897.

[32] C. Liu, Y. Wang, K. Kumar, and Y. Gong, "Investigation on speaker adaptation of lstm rnn models for speech recogntion," in *Proceedings of ICASSP*, Shanghai, China, Mar 2016.