Feature-Based Learning Hidden Unit Contributions for Domain Adaptation of RNN-LMs

Michael Hentschel*†, Marc Delcroix†, Atsunori Ogawa†, Tomohiro Nakatani†
* Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, Japan
michael.hentschel.mc5@is.naist.jp
† NTT Communication Science Laboratories, NTT Corporation, 2-4, Hikaridai, Seika-cho, Kyoto, Japan
{marc.delcroix, ogawa.atsunori, nakatani.tomohiro}@lab.ntt.co.jp

***Abstract*—In recent years, many approaches have been proposed for domain adaptation of neural network language models. These methods can be separated into two categories. The first is model-based adaptation, which creates a domain specific language model by re-training the weights in the network on the in-domain data. This requires domain annotation in the training and test data. The second is feature-based adaptation, which uses topic features to perform mainly bias adaptation of network input or output layers in an unsupervised manner. Recently, a scheme called learning hidden unit contributions was proposed for acoustic model adaptation. We propose applying this scheme to feature-based domain adaptation of recurrent neural network language model. In addition, we also investigate the combination of this approach with bias-based domain adaptation. For the experiments, we use a corpus based on TED talks and the CSJ lecture corpus to show perplexity and speech recognition results. Our proposed method consistently outperforms a pure non-adapted baseline and the combined approach can improve on pure bias adaptation.**

I. Introduction

Neural network language models (NN-LM) showed to outperform count-based language models (LM) in many tasks including automatic speech recognition (ASR). NN-LMs achieve a lower perplexity (PPL) as well as a lower word error rate (WER) when applied to N-best rescoring.

The domain adaptation of NN-LMs has been a popular task in recent years and many approaches have been proposed. These adaptation techniques can be separated into two main groups, namely model-based and feature-based adaptation. In model-based adaptation, many methods rely on introducing an adaptation layer in the model [1], [2], [3]. The model is first trained on general domain data, and the weights in the adaptation layer are re-trained with in-domain data. Another possibility has recently been introduced using a linear hidden network (LHN) [4], [5]. An LHN introduces another linear hidden layer without any subsequent non-linearity. The weights in the LHN are re-trained with in-domain data during model adaptation. For acoustic model adaptation in ASR, a concept called learning hidden unit contributions (LHUC) [6], [7] was described in previous research. LHUC introduces a multiplicative parameter for each unit of a hidden layer. The weights needed to calculate the adaptation parameters are learned on adaptation data. In LHUC, the adaptation weights are followed by a non-linearity, which limits the value range of the adaptation parameter between zero and two. This method has also been applied to model-based LM adaptation [8]. The adaptation weights of the LHUC are learned during re-training with in-domain data. In general, model-based adaptation methods require domain labels in the corpus for all the datasets used in training, validation and testing. Moreover, they require a relatively large amount of adaptation data [8]. In model-based adaptation, there is one model for every domain.

In addition to model-based adaptation, there have also been many methods proposed for feature-based domain adaptation. These methods make use of features to describe the topic of a text. A popular topic model with which to generate these features is latent Dirichlet allocation (LDA) [9]. LDA has been the most widely-used topic feature in previous research. The feature is used as an additional input to the network [10], [11], [12]. Through this additional input, topic features mainly add a domain dependent bias to the input of the network. The advantage of feature-based adaptation over model-based adaptation is that the topic features can be extracted in a completely unsupervised manner. This makes the approach applicable to a wider range of datasets, and it is in practice preferred to model-based adaptation. Moreover, all domains are covered by just a single model.

In this paper, we propose feature-based domain adaptation using LHUC and topic features generated from an LDA topic model. In our application of LHUC, the adaptation weights for the LHUC are calculated from topic features generated from an LDA. We apply LHUC adaptation parameters to the output of the LSTM. That means, it performs the domain dependent gating of the hidden units in the LSTM. In our experiments, feature-based LHUC showed a larger relative improvement than model-based LHUC.

In addition to feature-based LHUC domain adaptation, we also propose a combination with bias adaptation using an LHN. In our model, we use two different adaptation layers, one where the weights are adapted by LHUC and one where we add a topic dependent bias term. The outputs of both adaptation layers are added before the output layer. Our experimental results suggest that the effects of both adaptation schemes are complementary because the combined model further improves on bias adaptation or LHUC based adaptation. To the best of our knowledge, we are the first to apply this combination to LM domain adaptation.

All NN-LMs in the experiments use long short-term mem-

ory (LSTM) [13] in the recurrent layer because it is the current state of the art for recurrent neural network language models (RNN-LM) [14], [15], [16], [17]. The datasets we use for our experiments are based on TED talks and the corpus of spontaneous Japanese (CSJ) [18]. We use the Kaldi [19] based TED-LIUM [20] and CSJ recipes [21] to provide ASR results (100-best rescoring). For TED talks, we extended the training set for the LMs by including additional TED talks.

II. Recurrent Neural Network Language Model and bias-based Domain Adaptation

A. LSTM-LM

We first briefly review a baseline LSTM-LM as shown in Figure 1 (a). We encode the current word ID with a one-hot vector $\boldsymbol{w}(t)$. The input to the LSTM $\boldsymbol{x}(t)$ is calculated using the word embedding matrix $\boldsymbol{U}^{(\mathrm{w})}$

$$\boldsymbol{x}(t) = \boldsymbol{U}^{(\mathrm{w})}\boldsymbol{w}(t). \tag{1}$$

The following set of equations are used to calculate the output $\boldsymbol{h}(t)$ and the state $\boldsymbol{c}(t)$ of a single LSTM cell [15]

$$\boldsymbol{i}(t) = \sigma(\boldsymbol{W}^{(\mathrm{i,w})}\boldsymbol{x}(t) + \boldsymbol{W}^{(\mathrm{i,h})}\boldsymbol{h}(t-1) + \boldsymbol{b}^{(\mathrm{i})}), \tag{2}$$

$$\boldsymbol{f}(t) = \sigma(\boldsymbol{W}^{(\mathrm{f,w})}\boldsymbol{x}(t) + \boldsymbol{W}^{(\mathrm{f,h})}\boldsymbol{h}(t-1) + \boldsymbol{b}^{(\mathrm{f})}), \tag{3}$$

$$\boldsymbol{o}(t) = \sigma(\boldsymbol{W}^{(\mathrm{o,w})}\boldsymbol{x}(t) + \boldsymbol{W}^{(\mathrm{o,h})}\boldsymbol{h}(t-1) + \boldsymbol{b}^{(\mathrm{o})}), \tag{4}$$

$$\boldsymbol{g}(t) = \tanh(\boldsymbol{W}^{(\mathrm{g,w})}\boldsymbol{x}(t) + \boldsymbol{W}^{(\mathrm{g,h})}\boldsymbol{h}(t-1) + \boldsymbol{b}^{(\mathrm{g})}), \tag{5}$$

$$\boldsymbol{c}(t) = \boldsymbol{f}(t) \odot \boldsymbol{c}(t-1) + \boldsymbol{i}(t) \odot \boldsymbol{g}(t), \tag{6}$$

$$\boldsymbol{h}(t) = \boldsymbol{o}(t) \odot \tanh(\boldsymbol{c}(t)), \tag{7}$$

where $\boldsymbol{i}(t)$, $\boldsymbol{f}(t)$ and $\boldsymbol{o}(t)$ are usually named the input, forget and output gates, respectively. $\boldsymbol{W}^{(j,\mathrm{w})}$ and $\boldsymbol{W}^{(j,\mathrm{h})}$ denote the weight matrices for gate j for the word input and the previous hidden layer, respectively. The bias vectors for the respective gates are denoted by $\boldsymbol{b}^{(j)}$. Since we use vector notation, all the operators in the above equations are element-wise operators, which means $\sigma(\cdot)$ is the element-wise sigmoid, $\tanh(\cdot)$ is the element-wise hyperbolic tangent and $\odot$ is an element-wise multiplication.

$\boldsymbol{h}(t)$ is followed by a linear layer and the softmax function to calculate the probability for the next word $\hat{\boldsymbol{w}}(t+1)$

$$\hat{\boldsymbol{w}}(t+1) = \mathrm{softmax}(\boldsymbol{V}^{(\mathrm{w})}\boldsymbol{h}(t) + \boldsymbol{b}^{(\mathrm{V,w})}), \tag{8}$$

where $\boldsymbol{V}^{(\mathrm{w})}$ and $\boldsymbol{b}^{(\mathrm{V,w})}$ are the weight matrix and the bias vector, respectively.

B. Feature-Based Domain Adaptation with LHN

As a baseline for bias-based adaptation, we use domain adaptation with an LHN. This method was first proposed for the model-based adaptaion of vanilla RNN-LMs [5]. The authors used the LHN to add a domain dependent bias to the output of the RNN. The bias was given by the one-hot encoding of the domain label transformed by a linear layer as input into the LHN. In our case, however, there is no domain label given in the corpora used in our experiments and we want to focus on feature-based domain adaptation methods. To indicate the difference from the previously proposed model-based domain adaptation with LHN, we denote this method by fLHN. As adaptation features, we use features calculated from an LDA topic model.

Figure 1 (b) shows an fLHN. The LDA features $\boldsymbol{a}(t)$ are transformed by a linear layer with weight matrix $\boldsymbol{V}^{(\mathrm{a})}$ and bias vector $\boldsymbol{b}^{(\mathrm{V,a})}$. The output of the LSTM $\boldsymbol{h}(t)$ is also transformed by a linear layer with weight matrix $\boldsymbol{V}^{(\mathrm{w})}$ and bias $\boldsymbol{b}^{(\mathrm{V,w})}$. The outputs of these two linear layers are added at the input of the LHN

$$\boldsymbol{d}_{\mathrm{bias}}(t) = \boldsymbol{V}^{(\mathrm{w})}\boldsymbol{h}(t) + \boldsymbol{b}^{(\mathrm{V,w})} + \boldsymbol{V}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{V,a})}. \tag{9}$$

Equation (9) shows that an LHN adds a topic dependent bias term ($\boldsymbol{V}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{V,a})}$) to the output of the LSTM. The output of the LHN is followed by a linear layer $\boldsymbol{V}$ and the softmax function to calculate the probability for the next word $\hat{\boldsymbol{w}}(t+1)$

$$\hat{\boldsymbol{w}}(t+1) = \mathrm{softmax}(\boldsymbol{V}\boldsymbol{d}_{\mathrm{bias}}(t) + \boldsymbol{b}^{(\mathrm{V})}). \tag{10}$$

This is called a linear hidden network because there is no subsequent non-linear transformation. The LDA features are input into the LHN during network training and evaluation.

III. Proposed Feature-based Domain Adaptation with LHUC

Model adaptation with LHUC was first introduced for acoustic model adaptation in ASR. Recently, it has also been applied to vanilla RNN-LMs [8] and has been shown to reduce PPL and WER compared with a vanilla RNN-LM. In [8], the adaptation was applied as a model-based adaptation to the RNN. Because we focus on feature-based adaptation, we cannot use the conventional LHUC. Instead, we use the feature-based LHUC approach (fLHUC)[1]. This is the first time that the use of fLHUC has been proposed for LM adaptation. In our proposed approach, the adaptation weights are calculated from auxiliary features. We apply LHUC after a linear layer behind the LSTM as shown in Figure 1 (c).

$\boldsymbol{h}^{(\mathrm{a})}(t)$ is used as a gate for the output of the LSTM

$$\boldsymbol{d}_{\mathrm{LHUC}}(t) = (\boldsymbol{V}^{(\mathrm{w})}\boldsymbol{h}(t) + \boldsymbol{b}^{(\mathrm{V,w})}) \odot \boldsymbol{h}^{(\mathrm{a})}(t), \tag{11}$$

where $\odot$ denotes an element-wise multiplication of two vectors. The adaptation parameters in $\boldsymbol{h}^{(\mathrm{a})}(t)$ are calculated from the LDA features as follows

$$\boldsymbol{h}^{(\mathrm{a})}(t) = 2\sigma(\boldsymbol{U}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{U,a})}). \tag{12}$$

$\boldsymbol{U}^{(\mathrm{a})}$ and $\boldsymbol{b}^{(\mathrm{U,a})}$ are the weight matrix and bias vector of the linear layer for the LDA features. $\boldsymbol{h}^{(\mathrm{a})}(t)$ has values between $[0; 2]$ and it can be interpreted as the context dependent gating of the units in the adaptation layer. Because the sigmoid non-linearity will set certain nodes at zero, the authors in [6] used the weighting by a factor of two after the sigmoid. This should ensure that the activation on the input of each node in the following layer remains at the same magnitude. The output of

[1]For acoustic model adaptation this is also known as subspace LHUC [7].

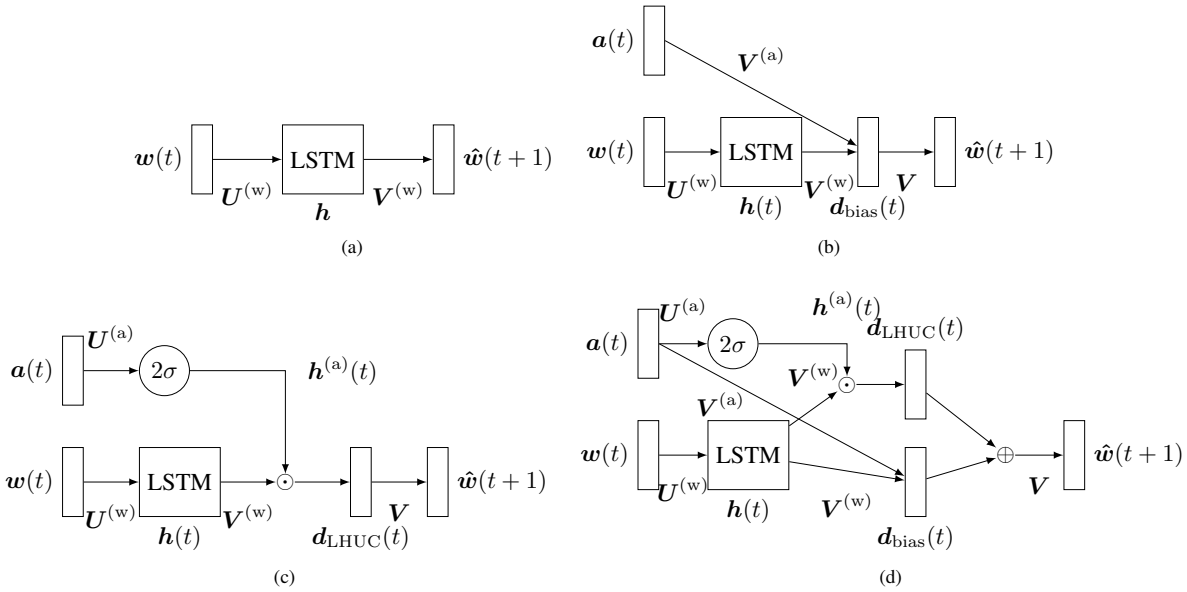

Fig. 1: (a) A conventional LSTM-LM, (b) LSTM-LM adaptation with linear hidden network (fLHN), (c) LSTM-LM adaptation with LHUC (fLHUC) and (d) LSTM-LM adaptation with LHUC and bias adaptation (fLHUCB).

the adaptation layer is followed by the output layer and the softmax function

$$\hat{\boldsymbol{w}}(t+1) = \mathrm{softmax}(\boldsymbol{V}\boldsymbol{d}_{\mathrm{LHUC}}(t) + \boldsymbol{b}^{(\mathrm{V})}). \quad (13)$$

The structure we use for the adaptation layer is identical to an LHN. An advantage of using an LHN rather than an adaptation of the LSTM output is that this layer can be used as a compression layer to reduce the number of parameters.

IV. Combination of LHUC and Bias-based Domain Adaptation for LSTM-LMs

The paradigm mainly used for NN-LM domain adaptation in the literature is bias adaptation. With the model shown in Figure 1 (d) we present an approach that combines bias adaptation and our proposed fLHUC. The model is a combination of the models described in Section II-B and Section III. We employ the addition of both adaptation layer outputs before the softmax function.

$$\begin{aligned}\boldsymbol{d}_{\mathrm{LHUC}}(t) &= (\boldsymbol{V}^{(\mathrm{w})}\boldsymbol{h}(t) + \boldsymbol{b}^{(\mathrm{V,w})}) \odot \boldsymbol{h}^{(\mathrm{a})}(t) \\ \boldsymbol{d}_{\mathrm{bias}}(t) &= \boldsymbol{V}^{(\mathrm{w})}\boldsymbol{h}(t) + \boldsymbol{b}^{(\mathrm{V,w})} + \boldsymbol{V}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{V,a})} \\ \hat{\boldsymbol{w}}(t+1) &= \mathrm{softmax}(\boldsymbol{V}(\boldsymbol{d}_{\mathrm{LHUC}}(t) + \boldsymbol{d}_{\mathrm{bias}}(t)) + \boldsymbol{b}^{(\mathrm{V})}).\end{aligned} \quad (14)$$

Bias adaptation has been shown to be effective in model- and feature-based domain adaptation methods and with fLHUCB we can also make use of this adaptation mechanism. As a result, by using fLHUCB we can combine the advantages of both bias adaptation and fLHUC in a single model.

V. Experiments

A. Dataset

1) TED Talks: In our experiments on the TED talks dataset, we used the TED-LIUM corpus [20] for ASR experiments. Our ASR system was based on the standard Kaldi [19] recipe. We trained a deep neural network acoustic model without any sequence discriminative training. To train the LMs, we used an enhanced dataset consisting of subtitles crawled from further TED talks. In total we crawled subtitles from 2494 talks. The resulting training set had a size of 5.1M tokens with a vocabulary size of 73K words. The vocabulary was thresholded to include only words that appeared more than once. This resulted in an effective vocabulary size of 43K words.

We used our own validation and test sets when training our LMs. We denote this evaluation set as the subtitle-based test set in this section. The order of the talks in the subtitle-based sets was the same as in the IWSLT 2011 evaluation campaign [22]. For consistency, we generated our datasets from the original subtitles in the same way as our 5.1M word training set. In the experimental result section, we will report results for the subtitle-based test set and the Kaldi recipe's test set.

To obtain verbatim transcriptions, the authors of the TED-LIUM corpus re-transcribed the talks for the validation and test sets. This introduced a mismatch with the subtitle-based set. A major difference is in the sentence length. As a result, the unigram probabilities of the end of sentence symbol 'EoS' are very different in the subtitle-based test set ($P(\mathrm{EoS}) = 0.1$) and in the TED-LIUM test set ($P(\mathrm{EoS}) = 0.04$).

2) CSJ: For the experiments with CSJ we also used the publicly available Kaldi recipe. The implementation for the acoustic model was the same as for the experiments with TED talks, namely Karel's DNN. Also for CSJ, we trained a deep neural network speech recogniser without any sequence discriminative training. The training data for our LMs were the same as those used for the LMs in the CSJ recipe. The training set had approximately 8.2M words. The vocabulary size was 71K and became 44K after retaining only words that appeared at least twice. The validation set we used during model training was the same as that used for training the LMs in Kaldi's CSJ recipe, namely the first 10K utterances of the training data.

B. LDA Training and Topic Estimation

To train our LDA topic model, we used each talk in the training sets as a separate document. Before training the LDA, we removed a list of common stopwords, and also high and low frequency words. We applied this preprocessing soley in order to train the LDA and to compute the LDA features. We used the LDA implementation in scikit-learn [23] for our experiments.

To calculate the LDA features for each word in the dataset, we used features from an LDA with 50 topics extracted from a sliding window covering the previous 200 words. The LDA features represent the topic distribution over this sliding window. In the ASR experiments, we calculated the LDA features from the 100-best lists. When the context window spans multiple utterances, we keep the context of the 1-best hypothesis in the context window.

C. NN-LM Training Parameters

The networks in our experiments were single-layer LSTMs with 300 units. The LHN also had 300 units. We trained all the networks for 40 epochs on TED and 20 epochs on CSJ. The results given below were chosen from the best model on the validation set. We chose a mini-batchsize of 128 and a backpropagation through time length of 20 words. The learning rate was 0.1 and we used the AdaGrad optimiser [24]. Gradients were clipped to an L2-norm of 5. In all our models, we applied dropout [25] with a dropout ratio of 50%. To implement our models, we used the open source toolkit Chainer [26].

D. TED Talk Results

For the TED talks, we chose to use a context window size of 200 words and an LDA size of 50. Table I shows our results for PPL and WER after 100-best rescoring. The results for the trigram were obtained with the trigram distributed with the Kaldi recipe [27]. All the parameters for rescoring, that is the language model weight and trigram interpolation weight, were tuned on the validation set.

fLHN reduced the PPL by 14% and 18% for the subtitle-based and TED-LIUM test sets, respectively, compared with the LSTM-LM baseline. The PPL we achieved with fLHN on the TED-LIUM test set was the lowest of all models we investigated. However, we did not observe any reduction in WER compared with the LSTM-LM baseline.

TABLE I: PPL and WER for our own subtitle-based test set and TED-LIUM. LDA features were calculated from an LDA with 50 topics and a 200-word window size. The trigram result represents the 1-best result and the results for the neural network LMs are for 100-best rescoring.

Model	Test		WER[%]	
	subtitle	TED-LIUM	val	test
trigram	156.41	222.05	16.3	15.1
LSTM	51.98	156.29	14.2	12.1
fLHN	44.72	**127.99**	14.1	12.1
fLHUC	46.98	135.68	14.0	11.9
fLHUCB	**38.65**	133.46	**13.9**	**11.8**

TABLE II: PPL and WER for CSJ using topic features from 50 LDA topics and a 200-word window size. The trigram result is the 1-best result and the neural network LM results are for 100-best rescoring.

Model	Test 1		Test 2		Test 3	
	PPL	WER	PPL	WER	PPL	WER
trigram	82.45	12.26	89.17	9.34	94.89	12.22
LSTM	40.52	10.71	41.79	8.08	41.01	10.49
fLHN	39.66	10.59	41.07	7.94	41.29	10.63
fLHUC	38.90	10.62	40.40	7.93	39.94	10.38
fLHUCB	**38.79**	**10.55**	**39.37**	**7.85**	**39.74**	**10.36**

With our proposed fLHUC, the PPL decreased relative by 10% and 13% for the subtitle-based test set and TED-LIUM test set, respectively. The WER reduction on the test set was 2% relative compared with the LSTM-LM baseline. The WER reduction with our fLHUC constitutes a greater improvement than realised for model-based LHUC adaptation in previous research and shows that LHUC-based domain adaptation can be successfully applied as a feature-based adaptation method.

The combination of bias and gating based domain adaptation in fLHUCB reduced PPL by 26% and 15% compared with the LSTM baseline. For the TED-LIUM test set, the PPL was between that of fLHN and fLHUC. This shows again the different characteristics of training and test data. We did observe a relative WER reduction of 2.5% compared with the LSTM-LM baseline on the test set. The PPL reduction on the subtitle based test set was the highest of all models. This shows that both methods have complementary information that helps to improve the performance in a matched setting.

E. CSJ Results

CSJ consists of a training set and three test sets, but there is no dedicated validation set. For parameter tuning for 100-best rescoring, we chose test 3 as a validation set. We used the same window size of the last 200 words and the same LDA size of 50 topics to calculate the LDA features as for the TED talk experiments. Table II shows the PPL and WER for all three test sets. Using an LSTM-LM reduced the trigram PPL by more than 50% and led to an improvement in 100-best rescoring.

With feature-based domain adaptation, we achieved an average relative PPL reduction of 3% compared with the LSTM-

LM baseline. This was much smaller than the 10% or higher relative reduction with the TED talks, but the baseline LSTM-LM already had a lower PPL for CSJ than for the TED talks. fLHN reduced the WER around 1% relative to the LSTM-LM in test 1 and test 2.

Our proposed fLHUC reduced the PPL by 3% to 4% relative to the LSTM-LM baseline. The relative WER reduction was between 1% to 2%. It achieved a consistent WER reduction across all test sets compared with the LSTM-LM baseline.

fLHUCB had the lowest PPL of all NN-LMs in all test sets. The PPL was reduced by 3% to 6% relative to the LSTM-LM baseline. WER was reduced relative by 1% and 3% in test 1 and test 2, respectively. Compared with the 1-best decoding result with a trigram LM, the WER was reduced 14%, 16% and 15% relatively for test 1, test 2 and test 3, respectively. The CSJ data are much better matched between training and test data than the TED talks dataset. Our numbers show that in this case fLHUCB achieved a consistent improvement over fLHN or fLHUC.

VI. Conclusion

We presented an approach for LHUC-based domain adaptation in NN-LMs in an unsupervised setting and showed its effectiveness in two common ASR tasks. With our application of LHUC for feature-based adaptation, we achieved a larger relative reduction in WER than in previous research. The method successfully benefited from topic information provided by an LDA topic model and consistently improved on the baseline LSTM-LM.

We also presented a simple approach for combining LHUC with bias adaptation. With matched training and test conditions, this method performed better than bias and LHUC based domain adaptation. This shows that bias and gating have complementary information that can help to improve performance.

Unlike earlier research, our approach does not need any domain annotation in the corpus and can therefore be applied to many corpora. In addition, since we use feature-based domain adaptation, we do not require a specific model for each domain and instead have only a single model covering all domains.

References

[1] J. Park, X. Liu, M. J. Gales, and P. C. Woodland, "Improved neural network based language modelling and adaptation," in *INTERSPEECH*, 2010.

[2] T. Alumäe, "Multi-domain neural network language model." in *INTERSPEECH*, 2013, pp. 2182–2186 vol. 13.

[3] O. Tilk and T. Alumäe, "Multi-domain recurrent neural network language model for medical speech recognition." in *Baltic HLT*, 2014, pp. 149–152.

[4] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. De Mori, "Linear hidden transformations for adaptation of hybrid ANN/HMM models," *Speech Communication*, vol. 49, no. 10, pp. 827–835, 2007.

[5] S. Deena, M. Hasan, M. Doulaty, O. Saz, and T. Hain, "Combining feature and model-based adaptation of RNNLMs for multi-genre broadcast speech recognition," in *INTERSPEECH*, 2016, pp. 2343–2347.

[6] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 171–176.

[7] L. Samarakoon and K. C. Sim, "Subspace LHUC for fast adaptation of deep neural network acoustic models." in *INTERSPEECH*, 2016, pp. 1593–1597.

[8] S. R. Gangireddy, P. Swietojanski, P. Bell, and S. Renals, "Unsupervised adaptation of recurrent neural network language models." in *INTERSPEECH*, 2016, pp. 2333–2337.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[10] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model." *SLT*, vol. 12, pp. 234–239, 2012.

[11] D. Soutner and L. Müller, "Application of LSTM neural networks in language modelling," in *International Conference on Text, Speech and Dialogue*. Springer, 2013, pp. 105–112.

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

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling." in *INTERSPEECH*, 2012, pp. 194–197.

[15] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 3, pp. 517–529, 2015.

[16] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv preprint arXiv:1312.3005*, 2013.

[17] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[18] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[19] 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," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011.

[20] A. Rousseau, P. Deléglise, and Y. Esteve, "TED-LIUM: an automatic speech recognition dedicated corpus." in *LREC*, 2012, pp. 125–129.

[21] T. Moriya, T. Tanaka, T. Shinozaki, S. Watanabe, and K. Duh, "Automation of system building for state-of-the-art large vocabulary speech recognition using evolution strategy," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 610–616.

[22] M. Federico, L. Bentivogli, P. Michael, and S. Sebastian, "Overview of the IWSLT 2011 evaluation campaign," in *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, 2011.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] J. Duchi, E. Hazan, and Y. Singer, "Adative subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[25] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[26] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Twenty-ninth Annual Conference on Neural Information Processing (NIPS)*, 2015.

[27] W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson, "Scaling recurrent neural network language models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015*. IEEE, 2015, pp. 5391–5395.