IEICE TRANS. INF. & SYST., VOL.E102–D, NO.3 MARCH 2019

PAPER

Feature Based Domain Adaptation for Neural Network Language Models with Factorised Hidden Layers

Michael HENTSCHEL[†*a)], **Marc DELCROIX**[††], ***Nonmembers***, **Atsunori OGAWA**[††], **Tomoharu IWATA**[††], ***and*** **Tomohiro NAKATANI**[††], ***Members***

SUMMARY Language models are a key technology in various tasks, such as, speech recognition and machine translation. They are usually used on texts covering various domains and as a result domain adaptation has been a long ongoing challenge in language model research. With the rising popularity of neural network based language models, many methods have been proposed in recent years. These methods can be separated into two categories: model based and feature based adaptation methods. Feature based domain adaptation has compared to model based domain adaptation the advantage that it does not require domain labels in the corpus. Most existing feature based adaptation methods are based on bias adaptation. We propose a novel feature based domain adaptation technique using hidden layer factorisation. This method is fundamentally different from existing methods because we use the domain features to calculate a linear combination of linear layers. These linear layers can capture domain specific information and information common to different domains. In the experiments, we compare our proposed method with existing adaptation methods. The compared adaptation techniques are based on two different ideas, that is, bias based adaptation and gating of hidden units. All language models in our comparison use state-of-the-art long short-term memory based recurrent neural networks. We demonstrate the effectiveness of the proposed method with perplexity results for the well-known Penn Treebank and speech recognition results for a corpus of TED talks.

key words: *language model, LSTM, domain adaptation, unsupervised, latent Dirichlet allocation*

1. Introduction

Language models (LMs) are usually trained on text data covering a large variety of domains. Domain specific language models, however, usually show a lower perplexity (PPL) and better performance in automatic speech recognition (ASR) tasks compared to general LMs at test time. Therefore, adapting general LMs to specific topics or genres has been an ongoing research interest. A comprehensive overview of N-gram LM adaptation techniques was provided in [1] and [2]. N-gram adaptation with topic model information was presented in [3]–[7]. N-gram LMs have been the most popular LMs in speech processing but there have also been several approaches of domain adaptation proposed for other LM types, such as, maximum entropy LMs [8]–[12].

Since the introduction of neural network based LMs (NN-LMs), these models have shown to perform consistently better than N-grams. NN-LMs were first introduced with feed-forward neural networks (FFWD-LMs) [13]. Subsequently, vanilla recurrent neural network LMs (RNN-LMs) [14], [15] further improved over FFWD-LMs, but suffered from the vanishing gradient problem [16]. In order to solve this problem, the vanilla RNN has been replaced with long short-term memory (LSTM) [17] to create LSTM-LM [18].

There exist two paradigms for domain adaptation of NN-LMs: model based and feature based domain adaptation. In model based adaptation, the parameters in the network are adapted with in-domain data by re-training. In feature based adaptation, an adaptation feature, such as, topic information from latent Dirichlet allocation (LDA) [19] is used. The LM is trained with the adaptation feature to adapt its activations in the network to the topic information. Compared to model based domain adaptation, feature based domain adaptation has the advantage that it does not require domain labels in the corpus. In addition, model based adaptation can have a problem with overfitting on little adaptation data [20].

We propose a new approach for feature based LM domain adaptation. Feature based domain adaptation methods usually add a bias to the network input or output. This bias depends on the topic feature or a (given) domain label. Our proposed factorised hidden layer based adaptation is fundamentally different from bias based approaches. The output layer is factorised into multiple layers or factors, where each factor is weighted by a factor weight. This can also be seen as a linear combination of different subspaces. It is also similar to a linear combination of multiple domain dependent LMs, which share a common hidden state. The factor weights are calculated from an auxiliary network, which is trained jointly with the main network by standard error backpropagation. The input of this auxiliary network are LDA features. Factorised hidden layer adaptation has been successfully applied to acoustic model adaptation [21] and speaker aware beamforming [22].

RNN-LMs have been the main target of domain adaptation techniques presented in the literature, but LSTM-LMs are currently the state-of-the-art. For this reason we will use LSTM as recurrent unit in all our NN-LMs. In addition, all adaptation methods we compare work in an unsupervised

Manuscript received June 22, 2018.
Manuscript revised October 11, 2018.
Manuscript publicized December 4, 2018.

[†]The author is with Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
[††]The authors are with NTT Communication Science Laboratories, NTT Corporation, Kyoto-fu, 619–0237 Japan.
[*]The work was conducted as joint research while the author was at NTT Communication Science Laboratories.
a) E-mail: michael.hentschel.mc5@is.naist.jp

DOI: 10.1587/transinf.2018EDP7222

manner, that means, these methods do not require any domain label in the training data. This setting is more relevant in practice because LMs can be trained on large text corpora of many million words. It would be costly and time intensive to annotate such large corpora by humans.

For our comparison, we use two different corpora. The first one is the well-known Penn Treebank (PTB) [23]. Despite its small size, PTB is one of the de-facto benchmarks in language modelling. As second corpus, we use a corpus based on TED talks. In order to provide ASR results, we use the TED-LIUM dataset [24], [25]. For language model training, we use our own enhanced training set, based on subtitles from more TED talks than used in TED-LIUM.

2. Overview of Domain Adaptation for Neural Network based Language Models

2.1 Model Based Domain Adaptation

Model based adaptation is a two-step process. First, a general language model is trained, where often an adaptation layer is inserted into the network. In the second step, the weights in this adaptation layer are updated using in-domain data. Model based adaptation has been used for FFWD-LMs [26], [27] and RNN-LMs [28]. Recently, model based adaptation with a linear hidden network (LHN) [29], [30] was proposed. An LHN adds a linear hidden layer in the network without a subsequent non-linearity. As input to this linear layer, the authors used the output of the RNN and a one-hot label which encodes the domain. The weights in this linear layer are learned during re-training.

A slightly different approach for model based adaptation uses a gating mechanism. In acoustic model adaptation, a concept called learning hidden unit contributions (LHUC) [31], [32] has previously been introduced. In this case, the speaker adaptation data is used to apply a gating mechanism on the hidden units in a neural network based acoustic model. In [20], the application of LHUC to RNN-LMs was investigated. The authors applied the concept of LHUC to the output of the hidden layer and the adaptation weights were learned from in-domain data. The authors showed improvements for PPL and N-best rescoring.

However, with little adaptation data, model based adaptation can be a problem because the adapted models are prone to overfitting [20]. In addition, model based adaptation requires domain labels throughout the whole corpus. Creating this annotation is expensive.

2.2 Feature Based Domain Adaptation

In feature based adaptation, usually the input of the network is extended by additional domain specific features. Many methods have been proposed, where these features act as a domain dependent bias. For RNN-LMs, the first proposed approach was a context dependent RNN-LM [33], which used LDA features to allow the network to exploit information from a context window of the current word. This method has also shown to be successful for RNN-LMs [34] on multi-domain broadcast data in the MGB Challenge [30]. The authors showed that feature based domain adaptation outperforms model based adaptation in the context of the MGB Challenge. Domain adaptation has mainly been proposed using vanilla RNN-LMs. To the best of our knowledge, the only other prior research on LSTM-LM adaptation was presented in [35], which uses the same adaptation mechanism as [33]. A feature based adaptation mechanism using a gating on the word vectors was proposed in [36]. The authors combined information from in-domain and general-domain word vectors.

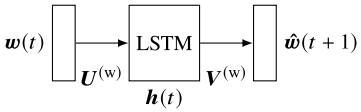

Fig. 1 A simple LSTM-LM.

3. LSTM Language Model

Before explaining more details about the adaptation techniques, we briefly review a baseline LSTM-LM as shown in Fig. 1. The vector encoding the current word ID by a one-hot vector is denoted by $\boldsymbol{w}(t)$. Using the embedding matrix $\boldsymbol{U}^{(\mathrm{w})}$, the input to the LSTM $\boldsymbol{x}(t)$ is calculated as follows,

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

To calculate its output $\boldsymbol{h}(t)$ and its state $\boldsymbol{c}(t)$, the following set of equations are used in an LSTM cell [37].

$$\boldsymbol{i}(t) = \sigma(\boldsymbol{W}^{(\mathrm{i,w})}\boldsymbol{x}(t) + \boldsymbol{W}^{(\mathrm{i,h})}\boldsymbol{h}(t-1) + \boldsymbol{b}^{(\mathrm{i})}), \tag{2}$$
$$\boldsymbol{f}(t) = \sigma(\boldsymbol{W}^{(\mathrm{f,w})}\boldsymbol{x}(t) + \boldsymbol{W}^{(\mathrm{f,h})}\boldsymbol{h}(t-1) + \boldsymbol{b}^{(\mathrm{f})}), \tag{3}$$
$$\boldsymbol{o}(t) = \sigma(\boldsymbol{W}^{(\mathrm{o,w})}\boldsymbol{x}(t) + \boldsymbol{W}^{(\mathrm{o,h})}\boldsymbol{h}(t-1) + \boldsymbol{b}^{(\mathrm{o})}), \tag{4}$$
$$\boldsymbol{g}(t) = \tanh(\boldsymbol{W}^{(\mathrm{g,w})}\boldsymbol{x}(t) + \boldsymbol{W}^{(\mathrm{g,h})}\boldsymbol{h}(t-1) + \boldsymbol{b}^{(\mathrm{g})}), \tag{5}$$
$$\boldsymbol{c}(t) = \boldsymbol{f}(t) \odot \boldsymbol{c}(t-1) + \boldsymbol{i}(t) \odot \boldsymbol{g}(t), \tag{6}$$
$$\boldsymbol{h}(t) = \boldsymbol{o}(t) \odot \tanh(\boldsymbol{c}(t)), \tag{7}$$

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

A simple LSTM-LM calculates the probability for the next word $\hat{\boldsymbol{w}}(t+1)$ from $\boldsymbol{h}(t)$ as

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

where $\boldsymbol{V}^{(\mathrm{w})}$ and $\boldsymbol{b}^{(\mathrm{V,w})}$ are the weight matrix and the bias vector of the output layer and softmax is the softmax function.

4. Feature Based Language Model Adaptation for LSTM-LMs

After reviewing the basic LSTM-LM, we introduce common

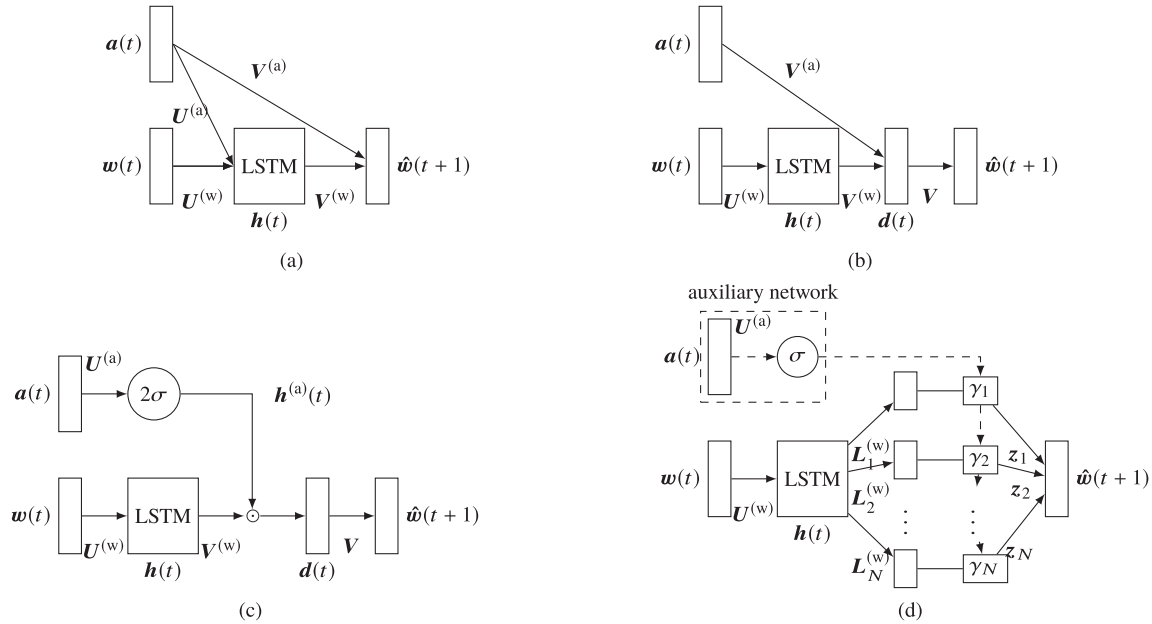

Fig. 2 LSTM-LM feature based model adaptation with (a) context dependent LSTM-LM (contLSTM), (b) LHN (fLHN-LSTM), (c) LHUC (fLHUC-LSTM) and (d) factorised hidden layer (factLSTM).

approaches for NN-LM domain adaptation. Although many of these approaches were introduced with vanilla RNN-LMs, we investigate their extension to LSTM-LMs. In addition, some of the approaches have been proposed for model based adaptation, however, we investigate their application to feature based adaptation. The features commonly used in prior research for domain adaptation were derived from an LDA topic model.

4.1 Context Dependent LSTM-LM

The first adaptation technique, as shown in Fig. 2 (a), has been originally proposed for RNN-LMs [33], [34] and has also been used with LSTM-LMs [35]. The input and output in a context dependent LSTM-LM (contLSTM) are dependent on the adaptation feature. The input is extended by an additional adaptation feature vector $\boldsymbol{a}(t)$. Equation (1) is modified as follows,

$$\boldsymbol{x}(t) = \boldsymbol{U}^{(\mathrm{w})}\boldsymbol{w}(t) + \boldsymbol{U}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{U,a})}, \tag{9}$$

where $\boldsymbol{U}^{(\mathrm{a})}$ and $\boldsymbol{b}^{(\mathrm{U,a})}$ are the weight matrix and bias vector for the adaptation features, respectively. There is also a direct connection introduced to the output layer which modifies Eq. (8) as follows,

$$\begin{aligned}&\hat{\boldsymbol{w}}(t+1)\\ &= \mathrm{softmax}(\boldsymbol{V}^{(\mathrm{w})}\boldsymbol{h}(t) + \boldsymbol{b}^{(\mathrm{V,w})} + \boldsymbol{V}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{V,a})}),\end{aligned} \tag{10}$$

where $\boldsymbol{V}^{(\mathrm{a})}$ and $\boldsymbol{b}^{(\mathrm{V,a})}$ are the weight matrix and bias vector of the linear layer connecting the adaptation feature vector to the output.

For RNN-LMs, the PPL reduction achieved with this adaptation scheme was significant compared to a vanilla RNN-LM. This can be explained to some extent by the vanishing gradient problem that RNNs suffer from [16]. The long context information provided by the adaptation features can circumvent this problem. In combination with LSTM-LMs, however, the relative PPL reduction achieved by this method was not as large as with vanilla RNN-LMs [35].

4.2 LHN Based LSTM-LM Adaptation

LM adaptation with a linear hidden network (LHN) has recently been proposed for model based adaptation of vanilla RNN-LMs [38]. Since we focus on feature based domain adaptation, we decided to use the LDA topic feature instead of a domain label as input to the LHN. In the experiments, we denote this method by fLHN-LSTM. Figure 2 (b) shows an fLHN-LSTM. The LHN introduces an additional linear layer between the LSTM and the output layer. Since the output $\boldsymbol{d}(t)$ of this intermediate linear layer is not followed by a non-linearity, it is called a linear hidden network.

The LDA features $\boldsymbol{a}(t)$ are transformed by a linear layer with weight matrix $\boldsymbol{V}^{(\mathrm{a})}$ and bias vector $\boldsymbol{b}^{(\mathrm{V,a})}$ and inserted into the LHN. The output of the LSTM $\boldsymbol{h}(t)$ is also transformed by a linear layer with weight matrix $\boldsymbol{V}^{(\mathrm{w})}$ and bias $\boldsymbol{b}^{(\mathrm{V,w})}$ and added at the input of the LHN

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

As shown in Eq. (11), LHN introduces a topic dependent bias term $(\boldsymbol{V}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{V,a})})$. The output of the LHN $\boldsymbol{d}(t)$ is followed by a linear layer $\boldsymbol{V}$ and the softmax function to calculate the probability for the next word $\hat{\boldsymbol{w}}(t+1)$

$$\hat{\boldsymbol{w}}(t+1) = \mathrm{softmax}(\boldsymbol{V}\,\boldsymbol{d}(t) + \boldsymbol{b}^{(\mathrm{V})}). \tag{12}$$

The LDA features are input to the LHN during network training and evaluation.

4.3 LHUC Based Domain Adaptation

Model adaptation with LHUC has first been introduced for acoustic model adaptation in ASR. Recently, it has also been applied to vanilla RNN-LMs [20] and it showed to reduce PPL and WER compared to a vanilla RNN-LM. In [20] the adaptation was applied as a model based adaptation of the RNN. We will, however, use LHUC based adaptation as a feature based adaptation method, where the adaptation weights are calculated from auxiliary features in a similar way to [32]. We use LHUC in a similar scheme to fLHN-LSTM, as shown in Fig. 2 (c). We will denote LHUC based model adaptation as fLHUC-LSTM.

The LDA features are multiplied with a linear layer, followed by a sigmoid non-linearity and a weighting by two (as proposed by [31], [32])

$$\boldsymbol{h}^{(\mathrm{a})}(t) = 2\sigma(\boldsymbol{U}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{U,a})}), \tag{13}$$

where $\boldsymbol{U}^{(\mathrm{a})}$ and $\boldsymbol{b}^{(\mathrm{U,a})}$ are the weight matrix and bias vector of the linear layer for the LDA features. The sigmoid non-linearity will set some of the activations to zero. In order to compensate for it, the amplitude of the remaining activations is multiplied by two, in order to keep the activation in the subsequent layer on the same level. $\boldsymbol{h}^{(\mathrm{a})}(t)$ is used as a gate for the output of the LSTM

$$\boldsymbol{d}(t) = (\boldsymbol{V}^{(\mathrm{w})}\boldsymbol{h}(t) + \boldsymbol{b}^{(\mathrm{V,w})}) \odot \boldsymbol{h}^{(\mathrm{a})}(t), \tag{14}$$

where $\odot$ denotes an element-wise multiplication of two vectors. $\boldsymbol{h}^{(\mathrm{a})}(t)$ has values between $[0, 2]$ and it can be interpreted as a context dependent gating of the units in the adaptation layer. Another linear layer and the softmax function then follow the output of this adaptation layer

$$\hat{\boldsymbol{w}}(t+1) = \mathrm{softmax}(\boldsymbol{V}\,\boldsymbol{d}(t) + \boldsymbol{b}^{(\mathrm{V})}). \tag{15}$$

We use this adaptation scheme, because in our experiments it showed to be more effective than an adaptation of the direct output of LSTM.

5. Proposed Factorised Hidden Layer Based LSTM-LM Adaptation

In this section, we introduce our proposed method for feature based domain adaptation using factorised hidden layers, as shown in Fig. 2 (d). We denote it hereafter as factLSTM. In our method, we use the output of the LSTM as an input to N linear layers (factors) with corresponding weight matrix $\boldsymbol{L}_n^{(\mathrm{w})}$ and bias $\boldsymbol{b}_n^{(\mathrm{L,w})}$. The size of each linear layer is the number of hidden units times the vocabulary size, that means the size of the output layer. After weighting each of the linear layers by a factor weight γ_n, they are summed up before the softmax function

$$\hat{\boldsymbol{w}}(t+1) = \mathrm{softmax}\left(\sum_{n=1}^{N} \underbrace{\gamma_n(\boldsymbol{L}_n^{(\mathrm{w})}\boldsymbol{h}(t) + \boldsymbol{b}_n^{(\mathrm{L,w})})}_{=z_n}\right). \tag{16}$$

As with the LHN, there is no non-linearity after the factors. There is only a multiplication with a factor weight before calculating the probability for the next word $\hat{\boldsymbol{w}}(t+1)$.

In order to calculate each factor weight γ_n, we use an auxiliary network. The input to the auxiliary network are the topic features calculated from an LDA topic model. This auxiliary network can be of arbitrary depth. In this work, we use a single linear layer followed by a sigmoid non-linearity

$$\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \ldots \gamma_n, \ldots, \gamma_N] = \sigma(\boldsymbol{U}^{(\mathrm{a})}\boldsymbol{a}(t) + \boldsymbol{b}^{(\mathrm{U,a})}), \tag{17}$$

where $\boldsymbol{U}^{(\mathrm{a})}$ and $\boldsymbol{b}^{(\mathrm{U,a})}$ are the weight matrix and bias vector for the linear layer. The parameters of the auxiliary network can be trained jointly with the main network by error back-propagation, as shown in [39]. This means we do not have to train the auxiliary network and the main network in separate training steps.

factLSTM is related to fLHUC-LSTM. In fLHUC-LSTM, one weight is multiplied with one node, whereas in factLSTM one weight is multiplied with one hidden layer. Comparing both methods, fLHUC-LSTM has the advantage that it needs less parameters, because it does not require multiple output layers. However, our proposed method has the advantage that individual factors can cover more domain specific information or information specific to certain domains.

6. Auxiliary Features for Feature Based Domain Adaptation

There exist many possible features for domain adaptation. The simplest one would be a one hot encoding of the domain. In prior research, topic information from an LDA topic model was the dominant feature for feature based domain adaptation. LDA provides a topic distribution over multiple topics contained in a document. That means, compared to a domain label, LDA provides a mixture of multiple topics and it is a continuous feature. Because LDA is such a common feature, we also use it in our experiments.

An important parameter to address for LDA is the specification of what should be regarded as a document. In case the training corpus consists of a single file, we regard a chunk of sentences as a single document for LDA training in the experiments. If the corpus can be separated into different talks, we regard one talk as a single document. In order to generate LDA features for each word, we estimate the topic distribution for a fixed size window of past words.

The LSTM itself captures some kind of context information, however, the cell state in the LSTM (which we relate to the context stored in an LSTM) is processed by an exponentially decaying function. That means, context further in the past has less influence on the current state than the more recent context. On the other hand side, LDA neglects word order and therefore all words in a document will

contribute equally when calculating a topic distribution.

7. Experiments

7.1 Dataset

7.1.1 Penn Treebank

For the experiments, we used two different datasets. First is the well-known PTB [23], which has roughly 0.9M words in the training set and a vocabulary size of 10K. The dataset consists of articles from the Wall Street Journal covering different topics, such as, politics and finance. We used the standard pre-processing, that is, sections 0-20 as training, 21-22 as validation and 23-24 as test set.

7.1.2 TED Talks

Our second corpus consists of TED talks and the TED-LIUM corpus [24] for our ASR experiments. For our ASR system, we used the standard Kaldi [40] recipe. The data provided in TED-LIUM is very small to train an LM. In order to have a larger training set, we crawled subtitles from other TED talks. Our final LM training set consisted of 2494 talks and had a size of 5.1M tokens with a vocabulary size of 73K words. We replaced every word which appeared only once in the training data by an unknown token. This resulted in an effective vocabulary size of 43K words.

We generated our own validation and test sets from the original subtitles in the same way as our 5.1M word training set. Our validation and test sets used the same data as in the IWSLT 2011 evaluation campaign [41]. We will report results for our subtitle test set and TED-LIUM's test set in the experimental result section.

The TED-LIUM validation and test set were re-transcribed from the original lectures in order to have verbatim transcriptions. This introduced a mismatch with the subtitle based sets. We summarised some key-figures about our subtitle-based and the TED-LIUM test set in Table 1. We further provide a histogram of the different sentence lengths in Fig. 3. As seen from Fig. 3, the sentence length for both evaluation sets is very different. As a result, the unigram probability of the end of sentence symbol is different in the subtitle based test set ($P(\text{EoS}) = 0.1$) and in the TED-LIUM test set ($P(\text{EoS}) = 0.04$).

7.2 LDA Training and Topic Estimation

For training our LDA model we applied two different schemes depending on the dataset. For PTB, we used the same processing scheme as in [15]. That means, we divided the training set in chunks of 10 utterances and each of these chunks was regarded as a single document. For the TED talks it was not necessary to make this segmentation because the dataset consists of different talks and we could use each talk as a separate document.

Before training the LDA, we removed a list of common stop words, as well as, high and low frequency words. This pre-processing has only been applied in order to train the LDA and to compute the LDA features. We did not apply this processing to the text corpora when training our LMs. The LDA implementation for our experiments was the one provided in scikit-learn [42].

In order to calculate the LDA features for each word in the datasets, we used LDA features extracted from a sliding window covering the previous 50 words in case of PTB and 200 words in case of TED. The LDA features represent the topic distribution over this sliding window. For N-best rescoring, we calculated the LDA features from the recognition result. In case the sliding window extended over the current utterance, it kept the content of the one-best result. Similarly, the state of the LSTM for the one-best hypothesis is kept across sentences.

Table 1 Comparison of subtitle and TED-LIUM test sets.

	sentence length (words)				Voc size	length (words)
	min	max	mean	var		
subtitle	2	19	9	8.88	3638	30168
TED-LIUM	2	122	25	209.39	3568	28655

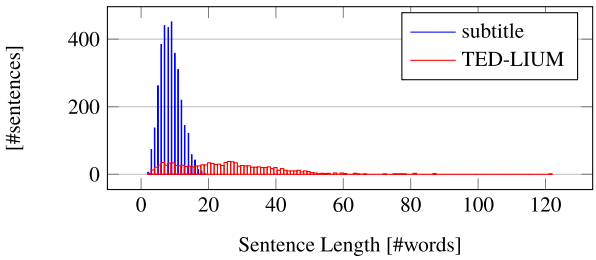

Fig. 3 Histogram of sentence lengths in subtitle and TED-LIUM evaluation sets.

7.3 Training Parameters for NN-LMs

As common part to all NN-LMs in the experiments, we used a single layer LSTM with 300 units. The LHN also had 300 units. For PTB, we trained all networks for 20 epochs and for TED we optimised the number of epochs for each model on the validation set WER. The mini-batchsize and backpropagation through time (BPTT) length were tuned on PTB to give a good compromise of training time and model PPL. We chose a mini-batchsize of 128 and BPTT length of 20 words. We set the initial learning rate to 0.1 and we used the AdaGrad [43] as optimiser. Gradients were clipped to an L2-norm of five. If the validation PPL improvement was less than 0.1% within one epoch, the learning rate was halved. In all our models, we applied dropout [44] with a dropout ratio of 50%. As for [38], in fLHN-LSTM, we initialised the weight matrix of the linear layer connecting the LSTM and the LHN ($\boldsymbol{V}^{(\text{w})}$) with the identity matrix. We implemented all NN-LMs with the open source toolkit Chainer [45]. We used Nvidia GTX 1080Ti GPUs with CUDA v8 and CUDNN v6 to train all NN-LMs.

Table 2 PPL on the validation set of PTB for different numbers of factorised hidden layers versus different LDA dimensions (LSTM-LM 105.66). The number in brackets with factLSTM gives the number of factors used.

Model \ LDA topics	30	40	50	60
contLSTM	118.72	117.83	122.72	121.74
fLHN-LSTM	103.99	105.59	107.16	105.76
fLHUC-LSTM	103.35	104.01	104.76	104.80
factLSTM (5)	105.06	105.55	105.93	106.08
factLSTM (10)	102.15	102.11	102.81	101.02
factLSTM (20)	102.92	102.80	101.54	101.06
factLSTM (30)	101.36	102.64	102.24	100.91
factLSTM (40)	103.75	102.69	101.75	**100.69**

Table 3 PPLs for baseline, best fLHN-LSTM and factLSTM model on the validation and test set of PTB.

Model	LDA topics	val	test
trigram	—	182.16	171.68
LSTM	—	105.66	98.94
contLSTM	40	117.83	109.00
fLHN-LSTM	30	103.99	97.42
fLHUC-LSTM	30	103.35	95.90
factLSTM (40)	60	**100.69**	**94.99**

7.4 Penn Treebank Results

At first, we show PPL results for PTB. For the domain adaptation methods, we analysed the behaviour with different numbers of LDA topics. Table 2 shows the results for LDA topic numbers from 30 to 60 for contLSTM, fLHN-LSTM, fLHUC-LSTM and factLSTM. contLSTM did not show any improvement over an LSTM-LM (validation PPL LSTM-LM: 105.66). This result is contrary to the one in [35], but the authors used only 20 LSTM units and we used 300 LSTM units in our experiments. In the case of a small number of hidden units, the LDA features might act as a context memory that the network can use. Another reason could be that it takes more epochs for this model to reach a lower PPL. fLHN-LSTM and fLHUC-LSTM were able to reduce the PPL compared to the LSTM-LM baseline in some cases.

The performance of factLSTM, depends on the number of LDA topics as well as the number of factors. Keeping the number of factors constant, in general the PPL decreased when the LDA size was increased. When the LDA size is kept constant and the number of factors is increased, the PPL was also reduced. However, with a small number of factors, the PPL is not much reduced compared to the LSTM-LM baseline. Also, choosing a large number of factors (larger than 10) did not yield much further PPL reduction on the PTB dataset. This might be due to the small size of PTB.

In Table 3 we show the best PPL results for each model. We estimated a baseline trigram LM with Kneser-Ney [46] smoothing using the SRILM toolkit [47]. The PPLs we show for the neural network LMs are, however, the values obtained using only the NN-LM without trigram interpolation. We used 40 LDA topics with contLSTM, 30 LDA topics with fLHN-LSTM and fLHUC-LSTM and 60 topics for factLSTM with 40 factors. The number of parameters were around 6.7M for contLSTM, around 6.5M for fLHN-LSTM and fLHUC-LSTM, and around 123M for factLSTM. The baseline LSTM-LM had around 6.4M parameters. The training time of LSTM-LM, contLSTM, fLHN-LSTM and fLHUC-LSTM was about 1h. factLSTM took 12h to train. contLSTM again had the overall highest PPL among the NN-LMs. fLHN-LSTM had a 1% lower PPL compared to the LSTM-LM baseline (relative improvement). fLHUC-LSTM reduced the PPL by 2% and 3% relative to the LSTM-LM baseline on the validation and test set, respectively. Overall, out of all domain adaptation methods, our proposed factLSTM had the highest relative PPL reduction compared to the LSTM-LM baseline. factLSTM improved 5% on the validation and 4% on the test set.

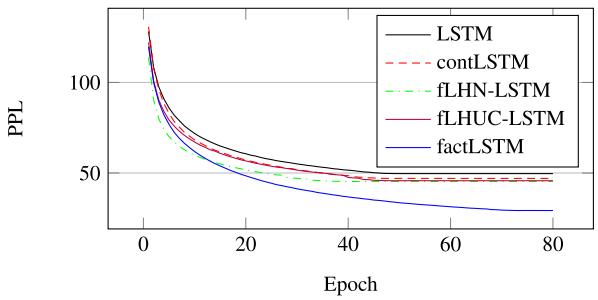

Fig. 4 Convergence on the validation set over 80 epochs for different models on TED dataset.

7.5 TED Talk Results

On the TED talks dataset, we trained each model until convergence according to the training scheme described in Sect. 7.3. The convergence of the validation PPL over 80 epochs is shown in Fig. 4. LSTM-LM reached its minimum after roughly 40 epochs. contLSTM showed a slight PPL reduction compared to LSTM-LM. fLHUC-LSTM had almost the same convergence as contLSTM. Both reached the minimum validation PPL around 50 epochs. fLHN-LSTM converged after around 40 epochs and convergence appeared to be faster than for LSTM-LM. factLSTM had the lowest PPL of all models after 20 epochs and reached the lowest PPL of all models at 80 epochs. The converged models had the lowest PPLs on the subtitle validation and test sets. However, PPL is not directly related with WER. In order to find the model with the lowest WER on TED-LIUM, we performed 100-best rescoring every five epochs around the point where the PPL converged. Subsequently, we chose for each adaptation method the model with the lowest WER on the validation set results and provide PPL and WER results for this model.

Table 4 shows our results for the TED-LIUM dataset. For the TED talks, we show PPL results for the trigram which is distributed with the Kaldi recipe [25]. The PPLs are, as for the PTB results, without any N-gram interpolation. We used the trigram to interpolate LM scores in 100-best rescoring. The corresponding interpolation weights for

each LM and the trigram were optimised on the TED-LIUM validation set. The interpolation weight of N-gram and NN-LMs was mainly between [0.9, 1] for the NN-LM. Therefore, the WER after rescoring was for the main part determined by the NN-LM. All NN-LMs with domain adaptation used LDA features from 50 topics and a window size of 200 words.

The PPL for the trigram is considerably higher than for the NN-LMs on the subtitle and TED-LIUM test sets. This can be explained by the fact that the trigram is trained on out-of-domain data. A trigram trained on our own training set had a considerably lower PPL of 106.58 on the subtitle test set. The baseline LSTM-LM reduced PPL by 67% compared to the trigram for the subtitle test set and 30% for the TED-LIUM test set. In 100-best rescoring, we decreased the WER by 20%. The baseline LSTM-LM had approximately 26M parameters and training for 40 epochs took approximately 4.5h.

contLSTM improved over LSTM-LM on the subtitle test set. On the TED-LIUM evaluation set, however, the PPL was slightly higher than the LSTM-LM. In 100-best rescoring, the WER was also slightly higher than the baseline. The number of model parameters increased by 2M to 28M and the training time increased by around 2h. With fLHN-LSTM PPL reduced by 14% and 18% compared to an LSTM-LM for the subtitle based and TED-LIUM test sets, respectively. In 100-best rescoring, the WER showed slight improvement over the LSTM-LM baseline for the validation set, but it was equal on the test set. fLHUC-LSTM had a slightly higher PPL than fLHN-LSTM on the subtitle based test set and TED-LIUM test set. The WER showed a slight but not significant reduction on the baseline. fLHN-LSTM and fLHUC-LSTM had 105K more parameters than an LSTM-LM, but the training time was about the same.

For factLSTM we used 15 factors due to the limitation of our GPU memory size. factLSTM had 207M parameters and the training time was 67h for 70 epochs. Our proposed method did increase the number of model parameters as well as the training time compared with the other methods, however, we did not perform any particular optimisation of the implementation to speed up the computation on GPUs. On the TED-LIUM test set, this method showed significantly lower PPL compared to all other methods, that is, 39% lower than LSTM-LM and 21% lower than fLHN-LSTM. In 100-best rescoring, the WER was 3% lower (relative reduction) compared to an LSTM-LM on the test set.

We performed a matched-pair significance test of all methods and it showed a significant difference of factLSTM compared with the LSTM-LM baseline at a significance level of 0.1%. All other methods did not show a significant improvement on the baseline. These results show that feature based adaptation can improve both PPL and ASR rescoring performance. Our proposed factLSTM achieved superior performance in general compared to other approaches for exploiting auxiliary features. This is however at the expense of using a larger amount of parameters.

Table 4 PPL and WER for our own subtitle based test set and TED-LIUM with 50 LDA topics, a 200-word window size and factLSTM with 15 factors. The trigram result represents the 1-best result and the results for the neural network LMs are for 100-best rescoring.

Model	Training Epochs	Test PPL		WER [%]	
		subtitle	TED-LIUM	val	test
trigram	—	156.41	222.05	16.3	15.1
LSTM	40	51.98	156.29	14.2	12.1
contLSTM	45	47.34	165.69	14.5	12.3
fLHN-LSTM	40	44.72	127.99	14.1	12.1
fLHUC-LSTM	40	47.69	144.70	14.0	11.9
factLSTM	70	**31.74**	**101.04**	**13.8**	**11.6**

8. Discussion

8.1 Impact of LDA Window Size

Following our main PPL and WER results on PTB and TED talks, we provide some further discussion on how LDA features can affect NN-LMs. At first, we discuss the features themselves. As we mentioned in Sect. 6, the LSTM itself captures context information and we believe that the LDA can capture different context information. In order to investigate when the information provided by the LDA can have a beneficial effect on the model performance, we modified the window length to calculate the LDA features.

Figures 5 and 6 show the validation set PPL for PTB and TED, respectively, where we modified the LDA window size. For PTB we chose window sizes from 50 to 500

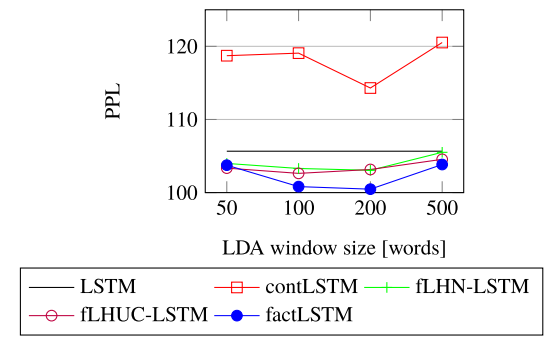

Fig. 5 Comparison of different context window sizes versus validation PPL for PTB with LDA features from 30 topics.

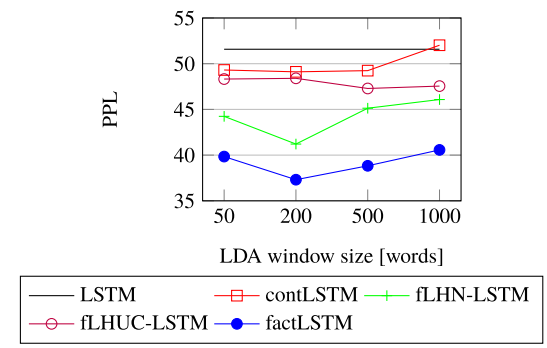

Fig. 6 Comparison of different context window sizes versus PPL for the Ted talks subtitle validation set with features from 50 LDA topics.

words. For TED we chose 50 to 1000 words because of the larger training data. For the ASR results in Sect. 7.5 we optimised the training time for each model to obtain the lowest possible WER. However, here we are only interested in PPL and consequently all NN-LMs in Fig. 6 were trained for 40 epochs.

For PTB the trend is that all models reached a minimum PPL with a window size between 100 and 200 words. Using a smaller or a longer window the PPL of domain-adapted models usually increased. This result indicates that shorter windows are well covered by the LSTM itself and longer windows contain too imprecise information. For TED also a context window size around 200 words gave a good result. In general, the LMs using domain adaptation improved on both corpora over a baseline LSTM-LM. This result suggests that there is information contained in the LDA features, which is not extracted by the LSTM itself.

Regarding the topic features themselves, during the experiments it showed that the adaptation performance is dependent on the quality of the topic model and the features estimated from it. Estimating a good LDA topic model will improve the feature quality and this is important for model performance. We believe that all models can benefit from better topic features, which might be derived from different topic models than LDA. Comparing the training set sizes of TED talks and PTB, for TED talks we have roughly five times more data. We think that this helped estimating better LDA topics, which were more effective for the LM and could lead to a larger PPL reduction.

8.2 Error Corrections after Rescoring

We conducted some more in-depth analysis of the recognition results before and after rescoring. Looking at the results for different model architectures, the NN-LMs had less errors with functional words. In the 1-best decoding result, these words were often left out or misrecognised. In cases with a large number of errors, rescoring with the NN-LMs did not reduce the number of errors considerably. In this case the output from the speech recogniser was the limiting factor. In some cases we were able to make a relation between the corrected error and a topic. We show one such example where we compare the ground truth utterance, the result after 1-best decoding, rescoring with LSTM-LM, and rescoring with factLSTM in Table 5.

We also investigated the probabilities each NN-LM assigned to the different hypothesises for one utterance more closely. Our findings did not show a constant offset in the probabilities between different models. We therefore conclude that each model architecture makes different use of the information provided by the LDA features.

8.3 Analysis of Correlation Between LDA Features and Factor Weights

For our proposed method factLSTM, we analysed what

Table 5 N-best hypothesis comparison for selected utterance from TED-LIUM test set.

REF:	this really solves the problem i've got a picture here of a place in kentucky this is the left over the ninety nine percent where they've taken out the PART THEY BURN NOW so IT'S called depleted uranium that would power the U S
1-best:	this really solves the problem i've got a picture here HAVE a place in kentucky this is the left over the ninety nine percent where they've taken out the **** **** PARTY BERNAU so **** called depleted uranium that would power the * U.S.
LSTM-LM:	this really solves the problem i've got a picture here of a place in kentucky this is the left over the ninety nine percent where they've taken out the **** **** PARTY BERNAU so **** called depleted uranium that would power the * U.S.
factLSTM:	this really solves the problem i've got a picture here of a place in kentucky this is the left over the ninety nine percent where they've taken out the part they burn now so it's called depleted uranium that would power the * U.S.

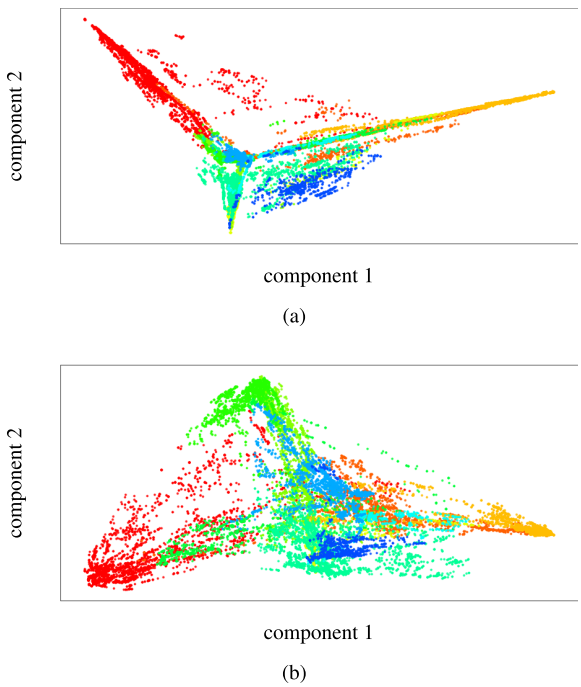

Fig. 7 PCA plots of LDA features (a) and factor weights (b) for the subtitle based TED talks test set. Each color represents a different talk in the test set.

factor weights the network learns from the LDA topic features. Figure 7 (a) shows a PCA plot of the LDA features for all talks in the TED subtitle based test set. Each colour represents one talk and one dot corresponds to one feature or factor weight, respectively. The plot reveals that the LDA features are mainly distributed along three axes. The PCA of the factor weights for the same talks is shown in Fig. 7 (b). The distribution is very different from the LDA feature plot. The network learns a mapping of the features, which helps to improve the prediction for the next word. In particular, some factor weights for different talks appear to be mapped to similar regions in the PCA space.

In addition to the PCA plots, Fig. 8 (a) and (b) show the LDA features and corresponding factor weights for part

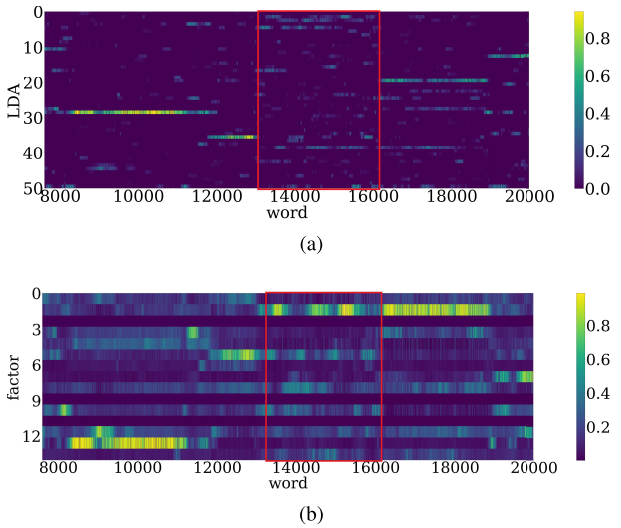

Fig. 8 Comparison of LDA features (a) and factor weights (b) for part of the test set (x axis corresponds to word index).

Table 6 Test PPL on TED talks for fLHN-LSTM with different LHN sizes after training for 70 epochs.

LHN size	300	600	1200
Parameters	26M	39M	65M
Training time	9h	11h	16h
subtitle PPL	45.25	41.98	42.79
TED-LIUM PPL	133.67	131.50	135.04

of the test set. As can be seen in the figures, there are distinct regions of high weights for the LDA features and high factor weights which correspond to each other. Interesting are the highlighted regions in Fig. 8 (a) and (b). In this interval, there is no distinct LDA feature with high intensity but factor one exhibits a high activation. The same factor keeps a high activation after word index 16000 which can be linked to one topic with high activation in this region. This is a good illustration how the auxiliary network learns to combine different topic features into a single factor weight. The corresponding factor can then learn common information among these different topics.

8.4 Parameter Size Comparison

Our proposed factLSTM has more parameters than the other models. We investigated if increasing the number of parameters in fLHN-LSTM can lead to a similar PPL as our proposed method. Table 6 shows PPL results for fLHN-LSTMs with different LHN sizes after 70 epochs for the subtitle based evaluation set of TED talks. We also give the number of parameters and the training time for each model. The PPL did not change considerably, even if we increased the LHN size four fold. As comparison, our proposed factLSTM had a PPL of 31.74 for the subtitle test set and 101.04 for the TED-LIUM test set. Our model had 207M parameters and the training time for 70 epochs was 67h. These numbers confirm that not only the increased number of parameters may cause the performance improvement of our proposed factLSTM.

9. Conclusion

We provided a comparison and discussion of different feature based domain adaptation techniques for NN-LMs and our proposed factLSTM. Feature based methods have in comparison to model based adaptation the advantage that they can be applied in an unsupervised manner. In practice, such methods are preferable because they do not require any domain information in the training, validation and test data which can only be made by experts. The LDA topic model we used in our comparison can also be trained in an unsupervised manner.

The methods we compared to our proposed one use two different strategies, that is, domain adaptation via an additive bias and a multiplicative gating function. Bias based adaptation with fLHN-LSTM was originally proposed for model based adaptation but, as the experiments showed, it can also be successfully applied in feature based adaptation. The PPL and WER were higher than factLSTM but it improved considerably over an LSTM-LM baseline. The advantage over factLSTM is that the number of parameters is much smaller and the convergence speed was the same as the LSTM-LM baseline on the TED talks dataset. Simple bias adaptation with contLSTM did not show a clear tendency to improve over an LSTM-LM in our experiments. Model adaptation with LHUC improved over the baseline LSTM, but achieved neither the same PPL as factLSTM nor a similar WER. On the datasets used in our experiments, our proposed approach factLSTM outperformed other methods for feature based domain adaptation.

References

[1] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?," Proc. IEEE, vol.88, no.8, pp.1270–1278, 2000.
[2] J.R. Bellegarda, "Statistical language model adaptation: review and perspectives," Speech communication, vol.42, no.1, pp.93–108, 2004.
[3] Y.C. Tam and T. Schultz, "Dynamic language model adaptation using variational bayes inference," Ninth European Conference on Speech Communication and Technology, 2005.
[4] A. Heidel, H.A. Chang, and L.S. Lee, "Language model adaptation using latent dirichlet allocation and an efficient topic inference algorithm," Eighth Annual Conference of the International Speech Communication Association, 2007.
[5] Y. Liu and F. Liu, "Unsupervised language model adaptation via topic modeling based on named entity hypotheses," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.4921–4924, IEEE, 2008.
[6] S. Watanabe, T. Iwata, T. Hori, A. Sako, and Y. Ariki, "Topic tracking language model for speech recognition," Computer Speech & Language, vol.25, no.2, pp.440–461, 2011.
[7] M.A. Haidar and D. O'Shaughnessy, "Topic n-gram count language model adaptation for speech recognition," Spoken Language Technology Workshop (SLT), pp.165–169, IEEE, 2012.
[8] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modelling," Computer Speech and Language, vol.10, no.3, pp.187–228, 1996.

[9] A.L. Berger, V.J.D. Pietra, and S.A.D. Pietra, "A maximum entropy approach to natural language processing," Computational linguistics, vol.22, no.1, pp.39–71, 1996.
[10] S. Della Pietra, V. Della Pietra, R.L. Mercer, and S. Roukos, "Adaptive language modeling using minimum discriminant estimation," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.633–636, IEEE, 1992.
[11] 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.
[12] T. Alumäe and M. Kurimo, "Domain adaptation of maximum entropy language models," ACL, pp.301–306, Association for Computational Linguistics, 2010.
[13] 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.
[14] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," INTERSPEECH, pp.1045–1048, 2010.
[15] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.5528–5531, IEEE, 2011.
[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE Trans. Neural Netw., vol.5, no.2, pp.157–166, 1994.
[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural computation, vol.9, no.8, pp.1735–1780, 1997.
[18] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," INTERSPEECH, pp.194–197, 2012.
[19] D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent Dirichlet allocation," Journal of machine Learning research, vol.3, pp.993–1022, 2003.
[20] S.R. Gangireddy, P. Swietojanski, P. Bell, and S. Renals, "Unsupervised adaptation of recurrent neural network language models," INTERSPEECH, pp.2333–2337, 2016.
[21] M. Delcroix, K. Kinoshita, A. Ogawa, C. Huemmer, and T. Nakatani, "Context adaptive neural network based acoustic models for rapid adaptation," IEEE/ACM Trans. Audio, Speech, Language Process., vol.26, no.5, pp.895–908, May 2018.
[22] K. Žmolíková, M. Delcroix, K. Kinoshita, T. Higuchi, A. Ogawa, and T. Nakatani, "Speaker-aware neural network based beamformer for speaker extraction in speech mixtures," INTERSPEECH, pp.2655–2659, 2017.
[23] M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The Penn Treebank," Computational linguistics, vol.19, no.2, pp.313–330, 1993.
[24] A. Rousseau, P. Deléglise, and Y. Esteve, "TED-LIUM: an automatic speech recognition dedicated corpus," LREC, pp.125–129, 2012.
[25] W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson, "Scaling recurrent neural network language models," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.5391–5395, IEEE, 2015.
[26] J. Park, X. Liu, M.J. Gales, and P.C. Woodland, "Improved neural network based language modelling and adaptation," INTERSPEECH, 2010.
[27] T. Alumäe, "Multi-domain neural network language model," INTERSPEECH, vol.13, pp.2182–2186, 2013.
[28] O. Tilk and T. Alumäe, "Multi-domain recurrent neural network language model for medical speech recognition," Baltic HLT, pp.149–152, 2014.
[29] 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.
[30] P. Bell, M.J. Gales, T. Hain, J. Kilgour, P. Lanchantin, X. Liu, A. McParland, S. Renals, O. Saz, M. Wester, et al., "The MGB challenge: Evaluating multi-genre broadcast media recognition," IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp.687–693, IEEE, 2015.
[31] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," Spoken Language Technology Workshop (SLT), pp.171–176, IEEE, 2014.
[32] L. Samarakoon and K.C. Sim, "Subspace LHUC for fast adaptation of deep neural network acoustic models," INTERSPEECH, pp.1593–1597, 2016.
[33] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," Spoken Language Technology Workshop (SLT), pp.234–239, IEEE, 2012.
[34] 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," INTERSPEECH, 2015.
[35] D. Soutner and L. Müller, "Application of LSTM neural networks in language modelling," International Conference on Text, Speech and Dialogue, pp.105–112, Springer, 2013.
[36] J. Zhang, X. Wu, A. Way, and Q. Liu, "Fast gated neural domain adaptation: Language model as a case study," International Conference on Computational Linguistics (COLING), pp.1386–1397, 2016.
[37] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," IEEE/ACM Trans. Audio, Speech, Language Process., vol.23, no.3, pp.517–529, 2015.
[38] 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," INTERSPEECH, pp.2343–2347, 2016.
[39] M. Delcroix, K. Kinoshita, C. Yu, A. Ogawa, T. Yoshioka, and T. Nakatani, "Context adaptive deep neural networks for fast acoustic model adaptation in noisy conditions," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.5270–5274, IEEE, 2016.
[40] 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 Workshop on Automatic Speech Recognition and Understanding (ASRU), IEEE, Dec. 2011.
[41] M. Federico, L. Bentivogli, P. Michael, and S. Sebastian, "Overview of the IWSLT 2011 evaluation campaign," International Workshop on Spoken Language Translation (IWSLT), 2011.
[42] 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.
[43] J. Duchi, E. Hazan, and Y. Singer, "Adative subgradient methods for online learning and stochastic optimization," Journal of Machine Learning Research, vol.12, pp.2121–2159, 2011.
[44] 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.
[45] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," Workshop on Machine Learning Systems (LearningSys) in the Twenty-ninth Annual Conference on Neural Information Processing (NIPS), 2015.
[46] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol.1, pp.181–184, IEEE, 1995.
[47] A. Stolcke, "SRILM – an extensible language modeling toolkit," International Conference on Speech and Language Processing, pp.901–904, 2002.

Michael Hentschel Michael Hentschel received the B.Sc. and M.Sc. degree in Information and Communication Technologies from Friedrich-Alexander University Erlangen-Nuremberg, Germany in 2013 and 2015, respectively. He is currently a Ph.D. candidate at Nara Institute of Science and Technology, Nara, Japan. His main research interests include speech signal processing, acoustic models and language models for automatic speech recognition.

Marc Delcroix Marc Delcroix received the M.Eng. degree from the Free University of Brussels, Brussels, Belgium, and the Ecole Centrale Paris, Paris, France, in 2003 and the Ph.D. degree from the Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan, in 2007. He was a research associate at NTT Communication Science Laboratories from 2007-2008 and 2010-2012 and then became a permanent research scientist at the same lab in 2012. He is also a visiting lecturer at the Faculty of Science and Engineering of Waseda University, Tokyo, Japan, since 2015. His research interests include robust multi-microphone speech recognition, acoustic model adaptation, integration of speech enhancement front-end and recognition back-end, speech enhancement and speech dereverberation. He is a member of the IEEE Signal Processing society Speech and Language Processing Technical Committee. He is a senior member of IEEE and a member of ASJ.

Atsunori Ogawa Atsunori Ogawa received the B.E. and M.E. degrees in information engineering, and the Ph.D. degree in information science from Nagoya University, Aichi, Japan, in 1996, 1998, and 2008, respectively. Since 1998, he has been with Nippon Telegraph and Telephone (NTT) Corporation. He has been engaged in researches on speech recognition and speech enhancement at NTT Cyber Space Laboratories and NTT Communication Science Laboratories. He is a member of the IEEE, International Speech Communication Association (ISCA), Institute of Electronics, Information, and Communication Engineers (IEICE), Information Processing Society of Japan (IPSJ), and Acoustical Society of Japan (ASJ). He received the ASJ Best Poster Presentation Awards in 2003 and 2006, respectively.

Tomoharu Iwata Tomoharu Iwata received the B.S. degree in environmental information from Keio University in 2001, the M.S. degree in arts and sciences from the University of Tokyo in 2003, and the Ph.D. degree in informatics from Kyoto University in 2008. In 2003, he joined NTT Communication Science Laboratories, Japan. From 2012 to 2013, he was a visiting researcher at Machine Learning Group, Department of Engineering, University of Cambridge, UK. He is currently a senior research scientist at NTT Communication Science Laboratories, Kyoto, Japan. His research interests include data mining and machine learning.

Tomohiro Nakatani Tomohiro Nakatani received the B.E., M.E., and Ph.D. degrees from Kyoto University, Kyoto, Japan, in 1989, 1991, and 2002, respectively. He is a Senior Distinguished Researcher (Supervisor) of NTT Communication Science Laboratories, NTT Corporation, Kyoto, Japan. Since joining NTT Corporation as a researcher in 1991, he has been investigating audio signal processing technologies for intelligent human-machine interfaces, such as dereverberation, denoising, source separation, and robust ASR. Dr. Nakatani was a Visiting Scholar of the Georgia Institute of Technology for a year from 2005 and he was a Visiting Assistant Professor in the Department of Media Science, Nagoya University from 2008 to 2017. He was honored to receive the 1997 JSAI Conference Best Paper Award, the 2002 ASJ Poster Award, the 2005 IEICE Best Paper Award, the 2009 ASJ Technical Development Award, the 2012 Japan Audio Society Award, the 2015 IEEE ASRU Best Paper Award Honorable Mention, and the 2017 Maejima Hisoka Award. He is a member of IEEE and ASJ.