MODELING NON-LINGUISTIC CONTEXTUAL SIGNALS IN LSTM LANGUAGE MODELS VIA DOMAIN ADAPTATION

Min Ma[⋆] *Shankar Kumar*[†] *Fadi Biadsy*[†] *Michael Nirschl*[†] *Tomas Vykruta*[†] *Pedro Moreno*[†]

[⋆] Graduate Center, The City University of New York, NY, USA mma@gradcenter.cuny.edu
[†] Google Inc., New York, NY, USA {shankarkumar, biadsy, mnirschl, tvykruta, pedro}@google.com

ABSTRACT

Language Models (LMs) for Automatic Speech Recognition (ASR) can benefit from utilizing non-linguistic contextual signals in modeling. Examples of these signals include the geographical location of the user speaking to the system and/or the identity of the application (app) being spoken to. In practice, the vast majority of input speech queries typically lack annotations of such signals, which poses a challenge to directly train domain-specific LMs. To obtain robust domain LMs, generally an LM which has been pre-trained on general data will be adapted to specific domains. We propose four domain adaptation schemes to improve the domain performance of Long Short-Term Memory (LSTM) LMs, by incorporating app based contextual signals of voice search queries. We show that most of our adaptation strategies are effective, reducing word perplexity up to 21% relative to a fine-tuned baseline on a held-out domain-specific development set. Initial experiments using a state-of-the-art Italian ASR system show a 3% relative reduction in WER on top of an unadapted 5-gram LM. In addition, human evaluations show significant improvements on sub-domains from using app signals.

Index Terms— neural network based language models, language model adaptation, domain adaptation, speech recognition

1. INTRODUCTION

Language models (LMs) play an essential role in automatic speech recognition (ASR). In a two-pass ASR system, two language models are typically used. The first is a heavily pruned n-gram model which is used to build the decoder graph efficiently. Another larger or more complex LM is employed to rescore hypotheses generated in the initial pass. In this paper, we focus on improving the second-pass LM. Recently, neural network based language models (NNLMs) have shown good performance in ASR [1, 2, 3]. An NNLM generally assigns a probability to a word w conditioned on the preceding words without incorporating non-linguistic contextual signals. However, contextual signals, such as application genre, geographical location or time period associated with speech query, and personal preference of speaker, contain rich information and may heavily influence language modeling [1, 4, 5, 6]. We explore a variety of contextual-signal based NNLM adaptation strategies for rescoring hypotheses generated by an ASR system.

In the context of ASR for voice search queries associated with a specific application ($a.k.a.$ app), recognition hypotheses employ a small subset of the recognizer vocabulary. For example, if a speech query is sent from *Android PlayStore*[1], it is more likely to contain a name or version of some game, rather than terms about say, weather or food. If a language model can capture application-specific information and target the next word within the domain, it is likely to make more accurate predictions. Nevertheless, in many cases, such signals are only available for a part of domain text but not for the generic text used in training an LM [6, 7]. The solution we propose to address the lack of contextual signals is to pre-train an LM on a large generic corpus, then adapt it to the given manually transcribed speech data whose app genre signals are available. We explore the adaptation of LSTM LM for short speech queries. We show that integrating contextual signals can help build powerful word-level LSTM LMs even for short utterances.

Pre-training and fine-tuning [8] remain as popular choices for domain adaptation of NNLMs [9, 6, 10]. However, deep neural networks are prone to catastrophically forgetting previous tasks [11]. We propose to categorize model parameters into general and domain parameters and tune them in different phases, suggesting a possible way to maintain the general performance of adapted LSTM LM.

2. PREVIOUS WORK

In [1], Mikolov and Zweig pointed out that feeding contextual signals as additional input to recurrent neural network (RNN) LMs is preferable for domain adaptation as it avoids building many individual signal-specific submodels. They extended the basic RNN LM with an additional contextual layer which was connected to both the hidden layer and output layer. By providing topic vectors associated with each word as inputs to the contextual layer, they obtained an 18% relative reduction in Word Error Rate (WER) on the Wall Street Journal ASR task. Chen et al. [4] performed multi-genre RNN LM adaptation by incorporating various topic representations as additional input features, which outperformed RNN LMs that were fine-tuned on genre-specific data. Experiments showed an 8% relative gain in perplexity and a small WER reduction compared to unadapted RNN LMs on broadcast news. Deena et al. [12] extended this work by incorporating a linear adaptation layer between the hidden layer and output layer, and only fine-tuning its weight matrix when adapting. They reported a 10% relative reduction in perplexity and a 2% relative reduction in WER, also on broadcast news.

Recently, Ghosh et al. [5] proposed an approach to use an embedding for topic dynamics and add it element-wisely to the calculation of gates and cell state of LSTM cells. Although the reduction of perplexity was small, the adapted LSTM LM showed a 21% relative improvement in accuracy for a sentence selection task on Wikipedia and an 18% relative improvement on Google News. Deena et al. [6] proposed four approaches to fine-tune RNN LMs with domain-specific features, and concluded that adding a separate weight matrix for the auxiliary features at the input layer gave the best performance. They derived auxiliary features from a multi-layer

[1] play.google.com/store

perceptron regression between the hidden state vectors of RNN LM and show-level LDA features. Their best adapted RNN LM showed a 16% relative reduction in word perplexity and a 1.3% relative reduction of WER when interpolating with a 4-gram LM in the second-pass to rescore 100-best lists.

3. DATA

We conduct all experiments on an Italian speech recognition task. We use a large corpus of texts consisting of aggregated and anonymized typed documents (42%), unsupervised spoken hypotheses (45%) and manually transcribed speech (13%), consisting of more than 260 million sentences (about 1.6 billion word tokens, no app signal annotations). We use the entire corpus as the training set of the background LM. A set of anonymized, unsupervised speech transcripts is split into two independent sets sharing the same domain distribution: domain adaptation set ($i.e.$ the training data in adaptation phase), which includes approximately 24 million sentences (about 110 million word tokens); domain development set, which is composed of about 6.8 million sentences (about 31 million word tokens). Each sentence of domain data is associated with an application label (Maps, YouTube, or PlayStore). We evaluate perplexities on an in-domain dev set and a held-out out-of-domain dev set, which contains 5.3×10^4 sentences (about 3.4×10^5 word tokens, no app annotations)[2]. We report the WER results for the top performing adapted LMs using a state-of-the-art ASR system on two dedicated ASR test sets. The modeled vocabulary contains 100k words.

4. LONG SHORT-TERM MEMORY LANGUAGE MODEL

Our baseline model is an RNN LM with two LSTM layers, each layer containing 1024 nodes. Each word is first represented by a 1-of-k encoding (all the out-of-vocabulary words are represented by an <unk> token) and then mapped to a 1024-dimensional embedding space. We train the LSTM LM with truncated back-propagation through time (BPTT) algorithm [13] with an unrolling of 20 time steps. Training loss is defined as the cross-entropy between predicted words and reference word labels. Mini-batch stochastic gradient descent (SGD) [14] is used with an Adagrad optimizer [15] and a batch size of 128 sequences. Learning rate of 0.2 is used. We found it crucial to use gradient clipping on the LSTM gradients (clipping L2-norm $\leq$ 1.0). In order to prevent models from overfitting, we employ dropout regularization [16] (with a keep probability of 0.8) to the input and output of the hidden layers for all LSTM LMs. We found it helpful to multiply gradients of word embedding layer with batch size, we apply this scaling to every model architecture unless its word embedding layer is *frozen* (cf. Sec. 5.1). We implement LSTM LM using augmented setting: we couple the input gate to 1.0 minus forget-gate to restrict the internal state of the LSTM cell to the unit interval, as in [17]; A peephole connection [18] from its internal cells to the gates in the same cell is used.

5. ADAPTATION APPROACHES

To better evaluate model performance, we set up three baselines (cf. Table1): 1) OOD-DATA ONLY BASELINE: model trained only on out-of-domain training data, without app signals. The trained LSTM LM shows a perplexity of 82 on domain dev set, and 85 on out-of-domain dev set. 2) DOMAIN-DATA ONLY BASELINE: model trained only on domain adaptation data, with app signals turned off. Its domain perplexity is 49 while out-of-domain perplexity is 293, which suggests a big difference between out-of-domain and in-domain data. 3) FINE-TUNING BASELINE: pre-train an LSTM LM on general training data, then fine-tune it on adaptation data, with app signals turned off. The adapted model reduces domain perplexity from 82 to 47, indicating that we can gain 42.7% relative reduction of domain perplexity by switching training data. However, its perplexity on out-of-domain dev set increased to 209, between the perplexities of first two baselines. The pre-trained LMs in the four approaches we propose are different, we compare their adapted domain performance with the domain perplexity of FINE-TUNING BASELINE.

5.1. Scheme I: Prepend App Id

Scheme I uses OOD-DATA ONLY BASELINE as background LM. In the adaptation, we prepend the word sequence with the app id and fine-tune the entire model. Intuitively, we assume initializing the LSTM state with contextual app ids might place the model in a better starting region, and proper embeddings of app ids will be learned by back-propagation. A conceptually similar idea was employed for image captioning using an RNN by Karpathy and Fei-Fei in [19], where the RNN state is initialized by convolutions of object regions in the image. As shown in Table 1, the scheme helps reduce domain perplexity to 39, which is 17.0% lower than the FINE-TUNING BASELINE. The 17.0% relative reduction in perplexity was obtained by only adding app signals.

In order to reduce the number of parameters to fine-tune, we apply a *freezing* technique [10] in the adaptation phase of Scheme I. In brief, we freeze some layers of the model, $i.e.$ we do not update the weight matrices of the frozen layers when back-propagating. We first freeze word embedding layer, aiming to leverage the learned representation of words. This adaptation strategy is "Variant 1" of Scheme I. Furthermore, we explore freezing both word embedding and LSTM layers as the "Variant 2". However, experiments show that neither of the freezing variants outperforms standard adaptation Scheme I in domain perplexity. It suggests when using "prepend" strategy, the best domain adaptation requires fine-tuning the entire model to adjust word embeddings and LSTM layers accordingly, although it results in worse performance on out-of-domain data than the freezing variants.

Table 1. *Perplexity of the Baselines and Scheme I (DOM PPL, OOD PPL refers to in-domain/out-of-domain perplexity. As OOD test results are noisy, a smoothing factor 0.6 is applied to raw values.).*

Adaptation Strategy	DOM PPL	OOD PPL
OOD-Data Only Baseline	82	85
Domain-Data Only Baseline	49	293
Fine-tuning Baseline	47	209
Standard Scheme I	**39**	211
Variant 1: freeze wEmb	41	176
Variant 2: freeze wEmb, LSTMs	46	153

5.2. Scheme II: Concatenate Input Embeddings

We add a separate embedding layer to encode the app id associated with each word sequence, then concatenate app embedding with

[2]The average number of words per sentence/utterance is 6.4 for training set, 4.6 for adaptation set, 4.6 for domain development set and 6.4 for out-of-domain development set.

word embedding at each time step and feed them to the hidden layers. The app signals are represented using 8-dimensional dense vectors. We feed zero vector as pseudo app signal for out-of-domain data during pre-training in order to keep pre-training and adaptation model architectures compatible. However, it is unlikely to learn useful information from an all-zero app embedding. This leads to a weaker background LM whose domain perplexity is 102.

In adaptation, we feed real app signals and fine-tune the entire model. The adapted model reduces the domain perplexity from 102 to 47 (cf. Table 2). However, when compared to FINE-TUNING BASELINE, feeding additional app signals by concatenation does not yield any gain in term of domain perplexity. One possible reason is that the background LM of "concat" scheme is much weaker than the one of FINE-TUNING BASELINE, which might limit the best adaptation performance it could achieve. More importantly, feeding app embedding along with every word might interfere with the LSTM learning by forcing model to continually see redundant information.

Two freezing variants are explored for Scheme II: 1) freezing word embedding layer reduces domain perplexity to 49, slightly worse than standard Scheme II. 2) freezing word embedding and LSTM layers yields even higher domain perplexity of 64. These results indicate that given the concatenation of word and app embeddings as input, it is hard to adapt well by only updating softmax output layer (and LSTM layers) of the model. We observe better out-of-domain performance of Scheme II than its freezing variants.

Table 2. *Domain and out-of-domain perplexity of Scheme II.*

Adaptation Strategy	DOM PPL	OOD PPL
Background LM (pseudo app)	102	1355
Standard Scheme II	**47**	163
Variant 1: freeze wEmb	49	249
Variant 2: freeze wEmb, LSTMs	64	332

5.3. Scheme III: Add Meta-Memory To LSTMs

We apply an affine transformation to app embedding when feeding it to LSTM cells, attempting to provide a persistent memory for each word sequence. We revise the formulae of LSTM from [17] to:

$$
\begin{aligned}
f_t &= \sigma(W_{fx}x_t + W_{fh}h_{t-1} + W_{fc}c_{t-1} + b_f + \mathbf{W_{fa}a} + \mathbf{b_{af}}) \\
i_t &= 1 - f_t \\
\hat{c_t} &= tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c + \mathbf{W_{ca}a} + \mathbf{b_{ac}}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \hat{c_t} \\
o_t &= \sigma(W_{ox}x_t + W_{oh}h_{t-1} + W_{oc}c_t + b_o + \mathbf{W_{oa}a} + \mathbf{b_{ao}}) \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

where $\odot$ refers to the element-wise product. The bold terms in equations indicate the affine transformation of app signals for domain data. These additional terms encode hidden topic information which word distribution might condition on, we call them "meta-memory". In pre-training, app embedding $\mathbf{a}$ is set to all-zero vector, and all new bias terms are initialized to random values close to zero. The domain perplexity is 83 for background LM, almost equal to OOD-DATA BASELINE. In the adaptation, real app embedding is fed and the entire model is fine-tuned. Scheme III achieved the best adapted domain perplexity of 37, which is 21.3% lower than FINE-TUNING BASELINE. However, its OOD perplexity of 602 is much higher.

To simplify Scheme III, we only incorporate meta-memory into the calculation of cell state candidate ($i.e.$ $\hat{c_t}$) for next time step t, not in the computation of the forget and output gates. We call this strategy "CandOnly Variant". It reduces domain perplexity to 39, slightly worse than 37 achieved by standard Scheme III. But the variant only fine-tunes a third of the number of parameters compared to Scheme III, and its out-of-domain perplexity of 276 is lower. We exploit various freezing settings for both adaptation strategies: 1) freeze word embedding layer only; 2) freeze word embedding layer, and the app-irrelevant weights and biases in LSTM cells (denoted as "LSTMs*" parameters). As shown in Table 3, freezing helps curb the model from shifting excessively towards domain data, resulting in lower out-of-domain perplexity but higher in-domain perplexity.

Table 3. *Domain and out-of-domain Perplexity of Scheme III.*

Adaptation Strategy	DOM PPL	OOD PPL
Background LM (Scheme III)	83	84
Standard Scheme III	**37**	602
Scheme III: freeze wEmb	41	176
Scheme III: freeze wEmb, LSTMs*	44	139
Background LM (CandOnly)	83	84
CandOnly Variant of Scheme III	**39**	276
CandOnly: freeze wEmb	48	177
CandOnly: freeze wEmb, LSTMs*	51	123

5.4. Scheme IV: Dual Paths To Softmax Output Layer

In the context of an ASR system, an adapted LM is expected to perform better on domain data, and maintain its performance on general (out-of-domain) data. However, perplexity evaluation shows that all the above schemes have moved the models too much towards domain data by fine-tuning most parameters, ending with reduced domain perplexity but increased out-of-domain perplexity. We note all these schemes feed app contextual signals to the layers below the softmax output layer. Hence in the adaptation, the entire model has to update to optimize cross-entropy on domain data. Upon completion of fine-tuning, adapted model tends to forget its knowledge of out-of-domain data learned in pre-training stage ($i.e.$ *catastrophic forgetting* [11]). Freezing lower layers helps control out-of-domain perplexity to some extent but is not able to yield better adapted domain perplexity.

Inspired by Biadsy et al. [7], we categorize parameters into two subsets: one for out-of-domain data, another for in-domain data. As our domain adaptation data is only 9% as large as out-of-domain training data, two problems will be introduced if we train the parameters jointly: in pre-training, there are no useful contextual signals to train domain parameters, which will lead to a weak background LM; more importantly, during adaptation, we actually prefer to keep the out-of-domain parameters to retain the performance on general data. We choose to tune the two types of parameters in different phases.

We start with pre-training an LSTM LM whose softmax output layer contains two kinds of parameters:

$$
P(w_t|hist) = \begin{cases} \phi(W_{OOD}h_t + b_{OOD}) & \text{Pre-training} \\ \phi(W_{OOD}h_t + b_{OOD} + W_D h_t{}' + b_D) & \text{Adaptation} \end{cases}
$$

where W_{OOD} and b_{OOD} stand for out-of-domain weights and biases of softmax layer, W_D and b_D for the domain ones. To preserve the knowledge learned from pre-training during adaptation, the out-of-domain parameters of the softmax layer and the layers below ($i.e.$ the LSTM and word embedding layers, denoted as "LSTM path") will be frozen. In pre-training, only the LSTM path is used and all the

out-of-domain parameters will be learned, while the domain parameters of output layer are initialized to zero and not updated. In the adaptation, we add a separate embedding layer to encode app signal, cascaded by a 1024-node neural network adaptation layer to connect to the same output layer (they form a "DNNadapt path"). We choose DNN rather than LSTM as adaptation layer as the app signal is not sequential. The word sequences are passed through the frozen LSTM path and the app signals are fed to the DNNadapt path separately. Domain parameters are learned on adaptation data. The outputs of two paths, h_t and $h_t{}'$ are combined at the softmax layer to make a final prediction of next word. This adaptation approach is "Scheme IV" ("dual paths"). Experiments (cf. Table 4) showed the adapted LSTM LM does maintain OOD perplexity of 85, just as its background LM, while reducing DOM perplexity to 75 (8.5% rel.).

The adapted domain perplexity of 75 is not as good as 47 obtained by the FINE-TUNING BASELINE, indicating only fine-tuning domain parameters of output layer and DNN path is not sufficient to obtain best domain perplexity. We then relax the freezing constraints to also tune OOD parameters of output layer, but at a slower pace by multiplying a factor of 0.25, 0.50, or 0.75. We note all the "LSTM path" is still frozen to prevent the model shifting excessively towards domain data. Fine-tuning OOD softmax parameters without penalty (equivalent to "mul 1.00") achieves the best domain perplexity of 51, but its OOD perplexity rises to 133. The performance reflect a trade-off between in-domain and out-of-domain perplexity.

Table 4. *Domain and out-of-domain perplexity of Scheme IV.*

Adaptation Strategy	DOM PPL	OOD PPL
Background LM (OOD Baseline)	82	**85**
Standard Scheme IV	75	**85**
Variant: mul 0.25	62	132
Variant: mul 0.50	61	152
Variant: mul 0.75	60	152
Variant: no penalty	**51**	133

6. EXPERIMENTAL RESULTS

6.1. ASR Experiment Results

Based on perplexity results, we select the top adapted LSTM LMs for the second-pass rescoring of a state-of-the-art Italian LVCSR system. The system employs a multi-pass rescoring framework: an unadapted Katz smoothed 5-gram LM is used for first-pass scoring, the first-pass lattice is rescored by adapted LSTM LM using the push-forward algorithm [20]. We linearly interpolate the two LMs using an interpolation weight of 0.5. We report performance on a short message dictation ASR task, with a domain WER baseline 13.2% and a general WER baseline of 12.9% (cf. Table 5). We find that fine-tuning pre-trained LSTM LM without app signals reduces domain WER from 13.2% to 12.9%; further reduction to 12.8% is achieved when incorporating app signals by prepending the word sequence with the app id (Scheme I). For the best adaptation strategy (Scheme I), we add 13 million random general utterances from the training set to the domain adaptation set ($i.e.$ a *mixed* adaptation set with 35% out-of-domain data). But it did not change the WER numbers, indicating that the "prepend" strategy is effective for addressing both general and domain data, thus it is not necessary to keep general data in the adaptation set. All adaptation strategies achieve lower domain WER relative to OOD-DATA BASELINE of 13.2%, demonstrating that the integration of app signals does adapt LSTM LM to specific domains. Adapted LSTM LMs maintain or slightly increase general WER.

Table 5. *WER (%) of ASR system based on test sets. General WER: evaluated on 8k test utterances (31% are domain data); Domain WER: evaluated on all the 3,015 utterances domain data only.*

Language Model	General WER	Domain WER
OOD-Data Only Baseline	12.9	13.2
Domain Data Only Baseline	13.4	13.2
Fine-tuning Baseline	12.9	12.9
Scheme I (prepend)	**12.9**	**12.8**
Scheme I: mixed data	**12.9**	**12.8**
Scheme II (concat)	13.0	12.9
Scheme III (meta-memory)	13.0	12.9
Scheme IV (dual paths)	13.0	12.9
Scheme IV: no penalty	12.9	13.0

6.2. SxS Experiment Results

To comprehensively evaluate our domain modeling approach, we run "side-by-side" (SxS) experiments [7], in which each anonymized test utterance is automatically transcribed by two ASR systems (FINE-TUNING BASELINE $vs.$ Scheme I). If the two transcripts differ, they will be presented to human raters. SxS experiments can accurately measure semantic changes as opposed to minor lexical differences. We conduct SxS experiments for each specific app domain, where we focus only on the fraction of the traffic affected by adapting to that domain. We present the following results: 1) Change: the percentage of utterances for which the two systems produced different transcripts. 2) Wins/Losses: the ratio of wins to losses in the experimental system vs. the baseline. 3) *p-value*, which is observed statistically significant (*p-value* $< 5.0\%$) for PlayStore and Maps, possibly because of the restrictive vocabularies in these domains. In future work, we will conduct SxS experiments on both in-domain and out-of-domain utterances to check if our adaptation scheme preserves the performance on both types of data.

Table 6. *SxS results for three app domains.*

Task	Win/Loss	% Change	*p-value*
PlayStore	58/36	2.4	0.1%-0.5%
Maps	82/49	0.9	1.0%-2.0%
YouTube	38/27	2.4	5.0%-10.0%

7. CONCLUSIONS AND FUTURE WORK

We demonstrate that an LSTM LM with contextual signals can obtain a 21% relative reduction in domain perplexity and a 3% relative reduction in WER on top of an unadapted 5-gram LM. Our SxS experiments show significant improvements on sub-domains from using app signals that provide complementary strength. Grouping the model parameters into two sets with freezing suggests a possible solution to obtain good performance on both in-domain and general data. In the future, we would like to further explore how to integrate multiple contextual signals ($e.g.$ location or time/date) from overlapping domains ($e.g.$ clustering) and generate more compact adaptation models ($e.g.$ embedding tying [21]).

8. REFERENCES

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

[2] Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran, "Deep neural network language models," in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Association for Computational Linguistics, 2012, pp. 20–28.

[3] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "LSTM neural networks for language modeling," in *Interspeech*, 2012, pp. 194–197.

[4] Xie Chen, Tian Tan, Xunying Liu, Pierre Lanchantin, Moquan Wan, Mark JF Gales, and Philip C Woodland, "Recurrent neural network language model adaptation for multi-genre broadcast speech recognition," in *Proceedings of InterSpeech*, 2015.

[5] Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck, "Contextual LSTM (CLSTM) models for large scale NLP tasks," *arXiv preprint arXiv:1602.06291*, 2016.

[6] Salil Deena, Raymond W. M. Ng, Pranava Madhyastha, Lucia Specia, and Thomas Hain, "Semi-supervised adaptation of rnnlms by fine-tuning with domain-specific auxiliary features," in *Proceedings of the INTERSPEECH*. 2017, ISCA.

[7] Fadi Biadsy, Mohammadreza Ghodsi, and Diamantino Caseiro, "Effectively building tera scale maxent language models incorporating non-linguistic signals," in *Proceedings of the INTERSPEECH*. 2017, ISCA.

[8] Geoffrey E Hinton and Ruslan R Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[9] Junho Park, Xunying Liu, Mark JF Gales, and Phil C Woodland, "Improved neural network based language modelling and adaptation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[10] Min Ma, Michael Nirschl, Fadi Biadsy, and Shankar Kumar, "Approaches for neural-network language model adaptation," in *Proceedings of the INTERSPEECH*. 2017, ISCA.

[11] Michael McCloskey and Neal J Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of learning and motivation*, vol. 24, pp. 109–165, 1989.

[12] Salil Deena, Madina Hasan, Mortaza Doulaty, Oscar Saz, and Thomas Hain, "Combining feature and model-based adaptation of RNNLMs for multi-genre broadcast speech recognition," in *INTERSPEECH*, 2016, pp. 2343–2347.

[13] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al., "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, pp. 1, 1988.

[14] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 661–670.

[15] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[16] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE, 2014, pp. 285–290.

[17] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[18] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber, "LSTM: A search space odyssey," *IEEE transactions on neural networks and learning systems*, 2016.

[19] Andrej Karpathy and Li Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3128–3137.

[20] Shankar Kumar, Michael Nirschl, Daniel Holtmann-Rice, Hank Liao, Ananda Theertha Suresh, and Felix Yu, "Lattice rescoring strategies for long short-term memory language models in speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2017 IEEE Workshop on*. IEEE, 2017.

[21] Ofir Press and Lior Wolf, "Using the output embedding to improve language models," *arXiv preprint arXiv:1608.05859*, 2016.