Active Memory Networks for Language Modeling

O. Chen, A. Ragni, M.J.F. Gales, X. Chen

Cambridge University Engineering Dept., Cambridge, U.K.
{ozhc2, ar527, mjfg, xc257}@cam.ac.uk

Abstract

Making predictions of the following word given the back history of words may be challenging without meta-information such as the topic. Standard neural network language models have an implicit representation of the topic via the back history of words. In this work a more explicit form of topic representation is used via an attention mechanism. Though this makes use of the same information as the standard model, it allows parameters of the network to focus on different aspects of the task. The attention model provides a form of topic representation that is automatically learned from the data. Whereas the recurrent model deals with the (conditional) history representation. The combined model is expected to reduce the stress on the standard model to handle multiple aspects. Experiments were conducted on the Penn Tree Bank and BBC Multi-Genre Broadcast News (MGB) corpora, where the proposed approach significantly outperforms standard forms of recurrent models in perplexity. Finally, N-best list rescoring for speech recognition in the MGB3 task shows word error rate improvements over comparable standard form of recurrent models.

Index Terms: language model, recurrent neural network, memory networks, attention, speech recognition, ASR

1. Introduction

Language models form a crucial component of many speech and language processing pipelines, such as in speech recognition and machine translation. In many state-of-the-art systems, a recurrent neural network language model (RNNLM) is combined with a n-gram language model [1] to obtain the best performance. The advantage of the RNNLM approach is the use of a long word history within a continuous hidden representation. Making accurate predictions of the following word given the back word history may be challenging without access to meta information. This limitation is particularly evident for tasks where the back word history is either ambiguous or can take on multiple meanings, such as in the case of topic modeling. The standard RNNLM only has an implicit topic representation via the back word history and may struggle to correctly learn this representation, unless provided explicit guidance [2, 3].

This paper introduces a novel recurrent network architecture, referred to as an Active Memory Network (AMN), that introduces a more explicit topic representation to the standard RNNLM structure via an attention mechanism. An AMN uses a recurrent attention mechanism to actively attend to K dynamic memory cells, where each memory cell may hold an *eigen* topic representation when trained to do so. At each time-step, an optimal topic representation for word-prediction can be obtained by interpolating the memory cells. Furthermore, a time-dependent regularization term is introduced to improve the training of an AMN. A high-level overview of the AMN architecture is shown in Figure 1, with further details in the following sections.

The model presented in this work is related to a range of attention and memory based models in the existing literature such as Neural Turing Machines [4] and memory networks for question answering [5, 6]. However, unlike previous works, a simpler approach is presented here for incorporating memory and attention into the model.

The rest of this paper is organised as follows. Section 2 gives a brief overview of neural network LMs. Section 3 presents the AMN architecture. The training and regularization methods are discussed in Section 4 and 5. Experimental results are presented in Section 6 with the Conclusion in Section 7.

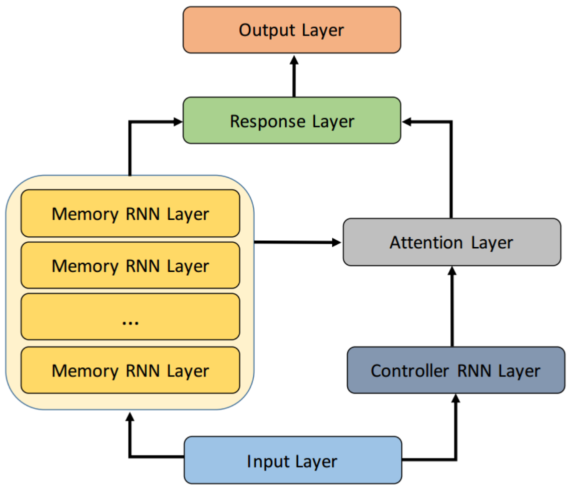

Figure 1: *Active Memory Network*

2. Neural Network Language Models

Language models (LMs) are generally classified as either discrete-space models such as n-grams, or continuous-space models such as neural networks [7]. Continuous-space models can be further divided between ones that assumes the Markov property to limit the word-history from $\mathbf{w}_{t-k}$ to $\mathbf{w}_t$ (feedforward neural networks) versus models that use the complete word-history from $\mathbf{w}_1$ to $\mathbf{w}_t$ (RNNLMs).

Unlike n-gram LMs, RNNLMs models the complete word-history by computing a continuous hidden vector $\mathbf{h}_t$ [8].

$$P(\mathbf{w}) = \prod_{t=1}^{T} P(\mathbf{w}_t|\mathbf{w}_{t-1}, ..., \mathbf{w}_1) \approx \prod_{t=1}^{T} P(\mathbf{w}_t|\mathbf{h}_t) \tag{1}$$

The hidden vector $\mathbf{h}_t$ is computed by:

$$\mathbf{x}_t = \mathbf{C}\mathbf{w}_t \tag{2}$$
$$\mathbf{h}_t = \tanh(\mathbf{W}_x\mathbf{x}_t + \mathbf{W}_h\mathbf{h}_{t-1}) \tag{3}$$

where $\mathbf{x}_t$ is the word-embedding for the word $\mathbf{w}_t$ obtained using an embedding matrix $\mathbf{C}$, $\mathbf{W}_x$ is the input-to-hidden weight matrix, and $\mathbf{W}_h$ is the hidden-to-hidden weight matrix. Computing the probability of the next word at time t is given by:

$$\hat{\mathbf{y}}_t = \text{Softmax}(\mathbf{W}_o \mathbf{h}_t) \tag{4}$$

where $\mathbf{W}_o$ is the hidden-to-output weight matrix, and $\hat{\mathbf{y}}_t = P(\mathbf{w}_t|\mathbf{h}_t)$ is the word probability distribution vector.

RNNLMs have been successfully applied in many language modeling applications [8, 9, 10, 11, 12]. Most state-of-the-art LMs are based on RNNs and their variants, such as GRUs or LSTMs [13, 14].

3. Active Memory Networks

As noted earlier, RNNs make use of a single hidden state to represent the input history, which implicitly includes any meta information. For language modeling, this meta information might correspond to the topics; for acoustic modeling, this meta information might correspond to speaker acoustic characteristics. This puts a lot of stress on a single hidden state, which may impair model training. Thus in some situations, it may be better to represent the meta information more explicitly. This would require specifying: (*i*) the model for representing the meta information and (*ii*) the method for training it. The Active Memory Network (AMN) is introduced as a first step in that direction, where a model that can learn to represent the meta information is provided along with a training method that requires no additional supervision.

At a high-level, an AMN uses a recurrent attention mechanism to attend over K parallel, time-dependent memory cells (abbrv. memcells) that share the same input $\mathbf{x}_t$. A diagram comparing an AMN to a RNN is shown in Figure 2. Unlike a RNN, an AMN contains multiple internal hidden states in the form of the memcell vectors $\{\mathbf{m}_t^{(i)}\}$. $\mathbf{m}_t^{(i)}$ can be interpreted as the hidden state of a single RNN, as shown by its input-to-hidden and hidden-to-hidden connections

$$\mathbf{m}_t^{(i)} = \tanh(\mathbf{W}_{x^{(i)}} \mathbf{x}_t + \mathbf{W}_{m^{(i)}} \mathbf{m}_{t-1}^{(i)}) \tag{5}$$

where $\mathbf{W}_{x^{(i)}}$ is the input-to-hidden weight matrix and $\mathbf{W}_{m^{(i)}}$ is the hidden-to-hidden weight matrix. Note that each memory cell has their own input-to-hidden weight matrix. Memory cells are selected by computing a soft-attention vector using a controller $\mathbf{u}_t$ implemented by

$$\mathbf{u}_t = \tanh(\mathbf{Q}_x \mathbf{x}_t + \mathbf{Q}_u \mathbf{u}_{t-1}) \tag{6}$$

where $\mathbf{Q}_x$ and $\mathbf{Q}_u$ are the input-to-hidden and hidden-to-hidden weight matrix respectively. To generate the attention vector

$$\beta_t^{(i)} = \mathbf{u}_t \cdot \mathbf{m}_t^{(i)} \tag{7}$$

is computed and passed through a softmax to obtain the attention weight $\alpha_t^{(i)}$ for $\mathbf{m}_t^{(i)}$:

$$\alpha_t^{(i)} = \frac{\exp(\beta_t^{(i)})}{\sum_j \exp(\beta_t^{(j)})} \tag{8}$$

A summation of the memory cell vectors weighted by the attention weights returns the response output $\mathbf{o}_t$.

$$\mathbf{o}_t = \sum_{i=1}^{K} \alpha_t^{(i)} \cdot \mathbf{m}_t^{(i)} \tag{9}$$

The predicted output at time t can then be computed by:

$$\hat{\mathbf{y}}_t = \text{Softmax}(\mathbf{W}_o \mathbf{o}_t) \tag{10}$$

Finally, the network can be trained using standard RNN algorithms, such as backpropagation-through-time (BPTT) [15].

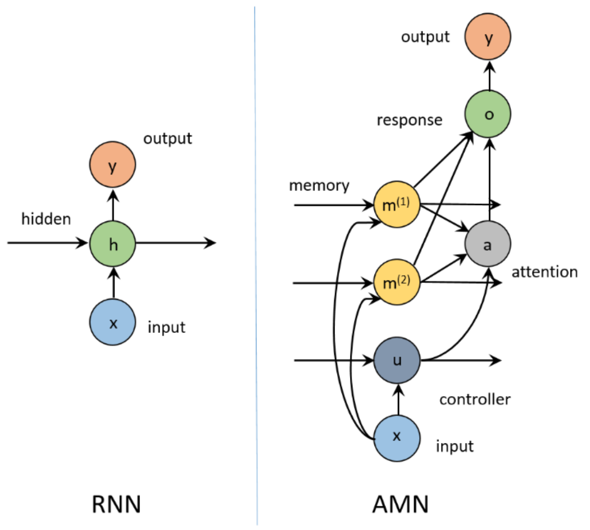

Figure 2: *Comparison of a RNN and AMN for a single time-step.*

4. Training the Attention Mechanism

Given the relatively complex architecture of the AMN model, training from random initialization using BPTT is unlikely to yield a robust attention mechanism. In the preliminary experiments, the model often converged to a trivial attention mechanism where many memcells were assigned near-zero attention weights. To understand this, it is useful to examine the derivative of the response output $\mathbf{o}$ with respect to the weights used to compute the memcell vector $\mathbf{m}^{(k)}$:

$$\frac{\partial \mathbf{o}}{\partial \mathbf{w}^{(k)}} = \alpha^{(k)} \frac{\partial \mathbf{m}^{(k)}}{\partial \mathbf{w}^{(k)}} \left(1 + \beta^{(k)} - \sum_{i=1}^{K} \alpha^{(i)} \beta^{(i)}\right) \tag{11}$$

Here, $\partial \mathbf{m}^{(k)} / \partial \mathbf{w}^{(k)}$ is the derivative of the k^{th} memcell vector with respect to its weights. Equation 11 implies that the weight-update for the k^{th} memory cell is directly proportional to the attention weight $\alpha^{(k)}$ assigned to it. In particular, when $\alpha^{(k)} = 0$, the k^{th} memcell never gets updated because the error gradient goes to 0. On the other hand, when $\alpha^{(k)} = 1$, only the k^{th} memcell gets trained due to the sum to one constraint.

4.1. Attention-weight annealing

One simple remedy to address such greedy training behavior is to force the model to activate all memcells during the first few training epochs. This can be enforced implicitly by using an annealing schedule

$$\alpha_t^{(i)} = \frac{\exp(\beta_t^{(i)}/T)}{\sum_j \exp(\beta_t^{(j)}/T)} \tag{12}$$

where T is the "temperature". As T approaches infinity, $\alpha_t^{(i)}$ approaches $1/K$, which implies that the attention is evenly distributed across all memcells. As T approaches 0, one of the weights $\alpha_t^{(i)}$ approaches 1 with the rest approaching 0, which implies that the model is focused on a single memcell. Thus T is initially set to a high value to encourage weight-tuning in all memcells, and slowly lowered at each training epoch by multiplying with $\gamma < 1$.

4.2. Dropout

Another technique that can prevent strong co-adaptation of attention weights is dropout [16, 17]. For the controller, this can be implemented using:

$$\mathbf{u}_t = \tanh(\mathbf{Q}_x(\mathbf{x}_t \odot \mathbf{z}_{u_t}) + \mathbf{Q}_u \mathbf{u}_{t-1}) \tag{13}$$

where $\mathbf{z}_{u_t}$ is a bit-mask vector sampled anew at each time-step. For the memcells, this can be similarly implemented by:

$$\mathbf{m}_t^{(i)} = \tanh(\mathbf{W}_x(\mathbf{x}_t \odot \mathbf{z}_{m_t^{(i)}}) + \mathbf{W}_{m^{(i)}} \mathbf{m}_{t-1}^{(i)}) \tag{14}$$

Figure 3 shows how each component of the model is affected by dropout in the controller and dropout in the memcells. For the memcells, the response output is changed by both the attention weights and the memcells. For the controller, the response output is only changed by the attention weights. This is shown by the dotted lines in the figure. This subtle difference in how dropout affects each model component leads to drastically different regularization behaviors. In particular, dropout in the memcells has the desirable property of regularizing both the memcell vectors and the attention mechanism. Note that

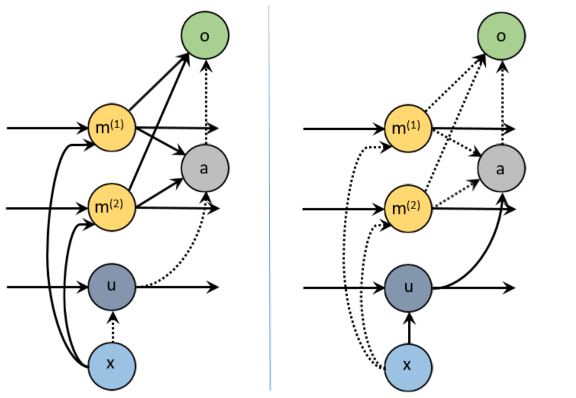

Figure 3: *Dropout in controller (left) versus dropout in memcells (right). Dotted lines indicates computation paths affected by noise injection from dropout.*

these regularization effects are due to the structural properties of AMN, and not the particular dropout method used. Thus, the regularization effects of dropout in the memcell will still hold even when an alternative method, such as [17], is used.

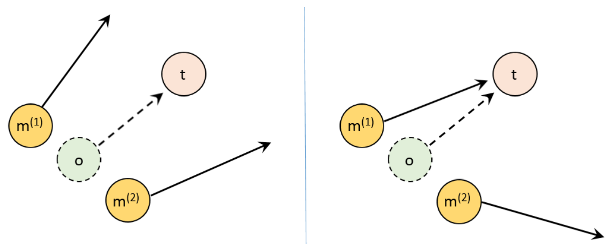

Figure 4: *Weight-update in original error function (left) and weight-update with ITL (right).*

5. Regularization with Implicit Target Loss

Given the use of attention and memory in the AMN model, a natural question which arises is: what should the memcells model? The original formulation of the model is quite unconstrained in that there are no training signals that dictates what the memcells should model. This can lead to a high-degree of co-adaptation in the memcells during training. An example is illustrated in Figure 4, where a weight-update pushes the response vector towards the desired target vector $\mathbf{t}$, even though the memcells are pushed towards different points. A reasonable prior in this situation is to have one of the memcells focus on modeling $\mathbf{t}$ and let the other memcell model some other vector.

To remedy this problem, a regularization term can be introduced to encourage memcell specialization during training:

$$R(\theta) = \lambda \sum_{i=1}^{K} \alpha_t^{(i)} \|\mathbf{o}_t - \mathbf{m}_t^{(i)}\|^2 \tag{15}$$

where $R(\theta)$ is the regularization term and λ is the tunable regularization penalty. This regularization term – referred to as the implicit-target loss (abbrv. ITL) – directly minimizes the loss between the memcell vector $\mathbf{m}_t^{(i)}$ and the response output $\mathbf{o}_t$. Intuitively, $\mathbf{o}_t$ provides a time-varying implicit-target for the memcells to directly model, where the quantity of the error contributed by each memcell is proportional to its attention value $\alpha_t^{(i)}$. The right hand side of Figure 4 shows how the weight-updates are changed by ITL. Instead of having both memcells weakly pushed towards $\mathbf{t}$, ITL induces a strong push towards $\mathbf{t}$ in one of the memcells, and allows the other one to wander.

In the special case of $R(\theta) = 0$, either (*i*) a single memcell is activated or (*ii*) the memcells are identical. Case (*ii*) is interesting since it suggests that ITL may encourage the model to train the memcells to be identical. However, if noise was injected into the computation of the memcells, it is highly unlikely that any of the memcells will ever be identical. In particular, the dropout technique for AMN discussed earlier will achieve precisely this effect. Consequently, $R(\theta)$ will be non-zero for most training cases when used in conjunction with memcell-dropout.

5.1. Interpreting ITL as MoE

The AMN model in equation (9) can also be viewed as a pseudo mixture-of-experts (MoE) [18, 19], with the gating function implemented using the controller and the memcells acting as the experts. However, unlike a typical MoE model, the memcell-experts do not directly model an output class-probability distribution $\mathbf{y}$. Instead, each memcell learns a hidden representation which is indirectly used to compute $\mathbf{y}$ via the interpolated response vector $\mathbf{o}$. To minimize the error, each memcell-expert needs to output a vector that is both useful for predicting $\mathbf{y}$ and accounts for the residual errors of the other memcell-experts. In this context, ITL can be interpreted as a way to de-couple the memcell interactions. Co-adaptation is discouraged because ITL will penalize the activation of memcells that are contributing a high residual error towards $\mathbf{o}$. This in turn frees the remaining memcells to further specialize for $\mathbf{y}$ without needing to account for the errors made by the non-specialized experts.

5.2. Interpreting ITL as L2 regularization

Alternatively, one can also interpret ITL as a form of L2 regularization on the memcells. This can be shown by setting $\mathbf{o}_t = \mathbf{0}$ and assuming that $\alpha_t^{(i)} = 1/K$.

$$R(\theta) = \tilde{\lambda} \sum_{i=1}^{K} \|\mathbf{m}_t^{(i)}\|^2 \tag{16}$$

where $\tilde{\lambda}$ is a scaled regularization penalty. This has implications for setting the initial weights, since smaller weights will likely push $\mathbf{o}_t$ towards $\mathbf{0}$ thus pushing ITL towards L2 regularization.

6. Experiments

Language modeling experiments were performed using the Penn TreeBank (PTB) dataset [16]. PTB consists mainly of text related to finance, politics and business. Speech recognition experiments were conducted on the BBC Multi-Genre Broadcast News (MGB) dataset [20] using an HTK [21] hybrid acoustic model trained on 275 hours of audio. MGB consists of both manually-transcribed and automatically-generated subtitles drawn from seven weeks of BBC broadcasts, and contains many genres/topics that can be learned by a LM.

All models were trained with roughly 16–20 million parameters to perform a fair comparison. The baseline RNN, GRU, and LSTM each had a single recurrent layer containing 750 hidden units. The AMN contained five memcells, where both the controller and the memcells were implemented using a GRU recurrent layer with 500 hidden units. Tensorflow [22] was used for all code implementation. More details on the experimental setup can be found in [23].

Model	Valid	Eval
RNN + Dropout	146	139
GRU + Dropout	115	114
LSTM + Dropout	127	117
AMN + Anneal + Drop-Mem	102	96
AMN + Anneal + Drop-Mem + ITL	98	**91**

Table 1: *PTB perplexity.*

Table 1 shows consistent perplexity deductions from the AMN model on PTB, with a relative test perplexity decrease of roughly 25% compared with the best performing baselines (GRU and LSTM). Training AMN with implicit target loss regularisation (ITL) resulted in the best performing model, believed to be due to the better regularization of the attention mechanism from ITL.

Model	Valid	Eval
KN-5 (Mikolov et. al 2012)	148	141
RNN + LDA (Mikolov et. al 2012)	132	126
TopicRNN (Dieng et. al 2017)	129	122
TopicGRU (Dieng et. al 2017)	118	112
TopicLSTM (Dieng et. al 2017)	126	118
AMN + Drop-Mem	108	103
AMN + Drop-Mem + ITL	104	97
AMN + Anneal + Drop-Mem + ITL	103	**95**

Table 2: *Comparison with explicit topic models on PTB.*

A comparison of AMN against models that perform *explicit* topic modeling is shown in Table 2. The AMN model for these experiments were trained with 100 recurrent units to compare with existing results. The perplexity results suggest that the topic modeling approach espoused by AMN provides a powerful alternative to the explicit topic modeling approach used by the other models. Moreover, the AMN modeling approach has the benefit of avoiding the need to set up the topic space, such as choosing the number of topics.

Perplexity and word error rate (WER) results on the MGB task are shown in Table 3. RNNLMs were trained on a smaller dataset due to computational constraints, which is why their perplexity results were worse than the tri-gram model – otherwise the perplexity results were similar to the results for PTB. Significant WER improvements were observed after interpolating with the n-gram LM for n-best rescoring – a common practice for speech recognition [8, 24, 25]. The best results were given by the AMNLM, which obtained a 0.2 WER absolute improvement over the GRU model. Figure 5 gives an example of the genre distribution among the memcells in MGB based on topic word-rankings. The genre word-ranking for memcell $\mathbf{m}^{(i)}$ was computed by ranking words in the genre-specific vocabulary according to their attention $\alpha^{(i)}$ averaged across all observations. The figure shows that at least one memcell was highly active for all genres. Note that no genre supervision was given to the AMNLM, so any genre specialization behaviors in the memcells were learned implicitly by the model.

Model	Datasets	Perplexity	WER
3-gram	Man+Sub	127	28.5
RNN	Man	198	27.8
GRU	Man	164	27.2
AMN	Man	142	27.0

Table 3: *MGB perplexity and WER from 100-best rescoring. All models were trained with 512 recurrent units.*

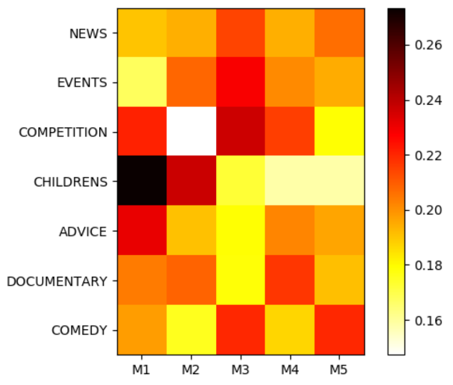

Figure 5: *Heatmap of genre distribution across memcells.*

7. Conclusions

Current forms of recurrent neural networks store an implicit representation of the meta information, such as the topic, in the back word history. In certain situations it may be advantageous to model such representation more explicitly. This paper proposed a model which stored multiple representations within memory cells and used a (soft) attention mechanism to select the most appropriate representation at each time-step. Since training these models from random initialization tend to lead the model to learn poor attention mechanisms, this paper introduced several training methods for yielding robust attention mechanisms. Experiments conducted on PTB and the MGB tasks show that this new model can outperform various standard recurrent language models in terms of perplexity and word error rate.

8. References

[1] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, “Class-based n-gram models of natural language,” *Computa-*

tional linguistics, vol. 18, no. 4, pp. 467–479, 1992.

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

[3] 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 *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[4] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[5] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[6] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[8] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

[9] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5528–5531.

[10] T. Mikolov, "Statistical language models based on neural networks," *Presentation at Google, Mountain View, 2nd April*, 2012.

[11] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[12] M. Sundermeyer, I. Oparin, J.-L. Gauvain, B. Freiberg, R. Schlüter, and H. Ney, "Comparison of feedforward and recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8430–8434.

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

[14] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

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

[16] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[17] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1019–1027.

[18] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[19] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[20] Y. Wang, X. Chen, M. J. F. Gales, A. Ragni, and J. H. M. Wong, "Phonetic and graphemic systems for multi-genre broadcast transcription," in *ICASSP*, 2018.

[21] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, A. Ragni, V. Valtchev, P. C. Woodland, and C. Zhang, *The HTK Book (for HTK Version 3.5)*. `http://htk.eng.cam.ac.uk`: University of Cambridge, 2015.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[23] O. Chen, "Memory networks for language modelling," Master's thesis, University of Cambridge, 2017.

[24] S. Kombrink, T. Mikolov, M. Karafiát, and L. Burget, "Recurrent neural network based language modeling in meeting recognition," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[25] X. Chen, A. Ragni, X. Liu, and M. J. Gales, "Investigating bidirectional recurrent neural network language models for speech recognition," 2017.