Persistence pays off: Paying Attention to What the LSTM Gating Mechanism Persists

Giancarlo D. Salton [*]
UNOESC
Campus Chapecó
Chapecó, Brazil
giancarlo.salton@unoesc.edu.br

John D. Kelleher
ADAPT Research Centre
Technological University Dublin
Dublin, Ireland
john.d.kelleher@dit.ie

Abstract

Language Models (LMs) are important components in several Natural Language Processing systems. Recurrent Neural Network LMs composed of LSTM units, especially those augmented with an external memory, have achieved state-of-the-art results. However, these models still struggle to process long sequences which are more likely to contain long-distance dependencies because of information fading and a bias towards more recent information. In this paper we demonstrate an effective mechanism for retrieving information in a memory augmented LSTM LM based on attending to information in memory in proportion to the number of timesteps the LSTM gating mechanism persisted the information.

1 Introduction

Language Models (LM) are important components in Natural Language Processing systems, such as Statistical Machine Translation and Speech Recognition (Schwenk et al., 2012). An LM is generally used to compute the likelihood of a sequence of words appearing in a given language. Recently, Recurrent Neural Networks LMs (RNN-LMs) have became the state-of-the-art approach to LMs (Józefowicz et al., 2016). However, RNN-LMs struggle to keep their level of performance as the length of the input increases.

A typical RNN-LM propagates a context vector that integrates information about previous inputs to use for the next prediction. Consequently, the information that is captured at the beginning of a sequence containing a long-distance dependency is likely to have faded from the context by the time the model spans that dependency. To address these limitations, several "memory-augmented" RNN-LMs architectures have been developed that attempt to retrieve relevant information from its past timesteps (e.g., Tran et al. (2016), Cheng et al. (2016), Daniluk et al. (2017), Merity et al. (2017), Grave et al. (2017) and Salton et al. (2017))

In this paper, we demonstrate that an efficient and effective mechanism for a memory augmented LSTM based LM (LSTM-LM) to retrieve important information from its history is to construct a representation of the LSTM unit state history that weights information in proportion to the number of timesteps the unit persisted the information. Using this strategy reinforces the decisions of the LSTM gating mechanism at each timestep regarding what is important in a sequence. Our models achieve competitive results on the Penn Treebank (Marcus et al., 1994) and on the wikitext2 (Merity et al., 2017). Structure: §2 presents the architecture of LSTMs; §3 discusses the effect of uniformly weighting the hidden states of an LSTM; §4 illustrates persistence of information in an LSTM and describes our memory augmented LSTM-LM; §5 presents experiments and results; §6 contextualizes our findings; and §7 our conclusions.

2 Long Short-Therm Memory

LSTM units (*aka.* LSTM cells) are now a normal building block for neural based NLP systems (Bradbury et al., 2017; Murdoch and Szlam, 2017). LSTMs retain and propagate information through the dynamics of the LSTM memory cell, hidden state and gating mechanism (including the *input*, *forget*, and *output* gates). The LSTM memory cell retains information that is only known by the unit itself and the hidden state shares informa-

[*] Work done while the author was at the ADAPT Center and the Dublin Institute of Technology.

https://doi.org/10.26615/978-954-452-056-4_121

tion to other LSTM units in the same or any next layer of the network. This way, the units can decide what to keep in memory and how much of that information it wants the other units/layers to know about it. If something is deemed important, the units will both keep it in memory and let other units/layers to know about it. The gating mechanism controls the flow of information between the memory cell and the hidden state. Therefore, the gating mechanism plays an important role on the LSTM hidden dynamics.

The computations of a standard LSTM unit (Gers et al., 2000) (without *peephole connections*) involve iterating over the following equations

$$\widetilde{\mathbf{c}}_t = tanh(\mathbf{W}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{(t-1)} + \mathbf{b}) \quad (1)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ii}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{(t-1)} + \mathbf{b}_i) \quad (2)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{if}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{(t-1)} + \mathbf{b}_f) \quad (3)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{io}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{(t-1)} + \mathbf{b}_o) \quad (4)$$

$$\mathbf{c}_t = \mathbf{f}_t \times \mathbf{c}_{(t-1)} + \mathbf{i}_t \times \widetilde{\mathbf{c}}_\mathbf{t} \quad (5)$$

$$\mathbf{h}_t = \mathbf{o}_t \times \tanh(\mathbf{c}_t) \quad (6)$$

where the weight matrices $\mathbf{W}_{i*}$ are associated to the input; the weight matrices $\mathbf{W}_{h*}$ are associated with the recurrence; the vectors $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the activation vectors produced by the *input*, *forget* and *output gates* respectively; $\widetilde{\mathbf{c}}_t$ is the candidate memory cell state; $\mathbf{c}_t$ is the new memory cell state; and $\mathbf{h}_t$ is the output of the unit.

The *candidate vector* (Eq. 1) contains information extracted from the input to the LSTM and, together with the *input gate vector* (Eq. 2) and *forget gate vector* (Eq. 3), is used to update the *memory cell* (Eq. 5). That update decides how much of the input is important to the memory cell, how much the *memory cell* will keep from its own content and what will be remembered in the memory cell for the next iteration.The *output vector* (Eq. 4) decides how much of the content in the memory cell $\mathbf{c}_t$ will be known on the next timestep (and by cells in the next layer if it is a multi-layered LSTM or to any layer that may come next o the network).

The success of LSTM-RNNs is attributed to their ability to retain information about the input sequence for several timesteps in their internal memory cell $\mathbf{c}_t$. That information is then made available to the next layer in the network for the amount of timesteps it is considered relevant to the current sequence. As pointed by Murdoch and Szlam (2017), each input to an LSTM makes a contribution to the hidden state of the LSTM and that is reflected when Eq. 5 is iterated. At any given timestep t, the cell state $\mathbf{c}_t$ can be decomposed into

$$\mathbf{c}_t = \sum_{i=1}^{t}(\prod_{j=i+1}^{t}\mathbf{f}_i)\mathbf{i}_i\widetilde{\mathbf{c}}_i \quad (7)$$

which, according to the authors, can be interpreted as the contribution at timestep t to the memory block $\mathbf{c}_t$ by a particular past input at timestep j. In that view, the contribution of an input to a given timestep can be understood as an importance score weighted by the LSTM's gating mechanism. Therefore, if something is important to the current context if should receive a larger importance score and be held in the memory block for a number timesteps. In addition to retaining information, Murdoch and Szlam (2017) have also demonstrated that, despite the fact that it is still difficult to interpret what specific activations in the hidden dynamics of LSTM units mean, it is possible to extract semantically meaningful rules from the memory cells to train a powerful classifier that can approximate the output of the LSTM itself. Moreover, Strobelt et al. (2016) and Karpathy et al. (2015) have demonstrated that these networks can extract meaningful attributes from the data into the memory cells. These attributes carry fine grained information and keep track of attributes such as line lengths, quotes and brackets.

Although these and other work demonstrate the power of LSTM units and their gating mechanism, RNN-LMs based on such units (LSTM-LMs) struggle to process long sequences. In our view, the main reason for this degradation in performance happens exactly because of the hidden state dynamics of the LSTM units. Once the information retained in the memory cell $\mathbf{c}_t$ is outdated, the *forget gate* $\mathbf{f}_t$ erases that block enabling the unit to store fresh data without interference from previous timesteps (Gers et al., 2000, 2003). This behaviour creates a natural bias towards more recent inputs given that the memory cell has limited capacity to store previous information and, once the memory cell is saturated, the *forget gate* will start to drop information in favour of more recent inputs. Even though the LSTM units can learn which information it must retain and for how long, the model will struggle with long sequences that are more likely to contain LDDs and that saturate the memory cell.

Once a memory cell has been saturated then, although some content has received a large importance score in past steps, it may be dropped from the memory cell (because of the inherent limitation of the LSTM's capacity of storing content) and will not be available to contribute to the next steps. For example, an LSTM-LM trained on English may persist the information related to a subject of a sentence for a number of time steps because the subject is important but this information may still have faded by the time the verb is reached. However, by augmenting the network with a memory buffer the information relating to the subject continues to be accessible so long as the memory buffer is not reset. This behaviour is an indication of why the memory augmented models such as the Neural cache model of Grave et al. (2017) and the Pointer LSTM of Merity et al. (2017) has gained success and achieved state-of-the-art results in LM research. Even though the required content has already faded from the context, the memory augmentation make it available for subsequent timesteps.

3 The Curious Effectiveness of Uniform Attention

As noted in Section 1, in recent years a number of extensions to RNN-LMs have been proposed to overcome the fading of information from context by adding a memory buffer (that is used to store the LSTM hidden states) and then at each timestep construct a representation of this history to inform the current prediction. A variety of relatively sophisticated mechanisms for retrieving information from the memory buffer have been proposed. In many cases these retrieval mechanisms include an extra neural network in the RNN-LMs that at each timestep predicts what elements in the memory buffer should be retrieved.

Salton et al. (2017) is a recent example that uses an extra neural network[1] to learn what to retrieve from memory. In this architecture at the end of a timestep the current LSTM hidden state is added to the memory buffer. At the beginning of the next timestep the additional neural network predicts an attention distribution over the elements of the buffer (*i.e.*, the previous LSTM hidden states). Using this distribution a compact representation of the RNN-LMs history is constructed by calculating a weighted sum of the elements in the memory (where the weight of each element is the attention attributed to it by the RNN). Curiously, although this architecture was successful in terms of performance the attention mechanism did not work as expected. Instead of focusing attention for each time step on particular relevant elements in memory it spread out the attention nearly uniformly across the memory. It might appear that this architecture was using a strategy of "pay equal attention to everything in the past". However, we argue this interpretation ignores the power of the LSTM gating mechanism.

Our interpretation of the uniform attention mechanism presented by Salton et al. (2017) is that their *Attentive* RNN-LM is in fact (indirectly) reinforcing the decisions of the gating mechanism of the LSTM units and is retrieving information that is persisted across multiple timesteps. This is important because it indicates that it may be more fruitful and efficient to leverage the decisions made by the LSTM gating mechanism (decisions that the network must make anyway) to drive the retrieval of information from the memory buffer rather than train a separate neural network. It is worth emphasising that to date none of the different retrieval mechanisms proposed in the literature on memory augmented LSTM-LMs have explicitly considered the behaviour of the LSTM gating mechanism.

4 The Persistence of Information

The LSTM gating mechanism will attempt to persist important information for as long as possible (or until the state is saturated). We propose that when retrieving/constructing a representation of the LSTM history from a memory buffer the information held for more than one timestep should be weighted in proportion to the number of timesteps the LSTM gating mechanism persisted it across. This way, we let the gating mechanism of the LSTM determine what is important about the input and, anything that is persisted for more than one timestep, will have a greater impact on the final prediction even if that information has already faded from the current context.

A simple and efficient way to implement this strategy is at each time point to construct a representation of the history of the RNN-LM that is simply an average of the LSTM hidden states in

[1]Similar to that proposed by Bahdanau et al. (2015) and Luong et al. (2015) for Neural Machine Translation (NMT)

the memory buffer. Pieces of information that the LSTM unit persists for several time steps will have a bigger impact on this average (simply because they are included multiple times) relative to items that are not persisted. In effect, this average weights each piece of information in proportion to the number of time steps the LSTM persisted it and so an RNN-LM that uses this average as its representation of history pays attention to what the LSTM gating mechanism persisted.

4.1 Averaging the Outputs

In this work we simplify the architecture of Salton et al. (2017) and use an average of previous outputs instead of a neural network based attention mechanism. Our intuition for this modification is that the gating mechanism of the LSTM is telling us what is important about an input and that we must find a way to make that information available for long distances in the future. In fact, Ostmeyer and Cowell (2017) have presented a model that computes a recurrent weighted average (RWA) over every past hidden state. However, the authors limit themselves to evaluate the model over simple tasks and the effectiveness of that model over language modelling is still to be demonstrated.

Compared to other memory augmented models our architecture is relatively simple. A multi-layered LSTM-RNN encodes an input at each timestep and the outputs of the last recurrent layer (*i.e.*, its hidden state called h_t) is added to memory. At each timestep an average of the vectors in the memory buffer is calculated and concatenated with the h_t generated by the processing of the current input. This concatenated vector is then feed into the softmax layer which predicts the distribution for the next word in the sequence.

In our experiments with this uniform attention, we found that initialising the memory with a zero vector $\mathbf{h}_0$ and allowing the model to count this vector as part of the memory when calculating the average[2] improved the performance of the model.

5 Experiments

To test our intuitions, we evaluate the averaging process of the model using the PTB dataset using the standard split and pre-processing as in Mikolov et al. (2010) which consists of 887K, 70K and 78K tokens on the training, validation and test sets respectively. We also evaluate the model on the wikitext2 dataset using the standard train, validation and test splits which consists of around 2M, 217K tokens and 245k tokens respectively.

5.1 PTB Setup

Following Salton et al. (2017) we trained a multilayer LSTM-RNN with 2 layers of 650 units for the PTB experiment. We trained them using Stochastic Gradient Descent (SGD) with an initial learning rate of 1.0 and we halved the learning rate at each epoch after 12 epochs. We train the model to minimise the average negative log probability of the target words until we do not get any perplexity improvements over the validation set with an early stop counter of 10 epochs. We initialize the weight matrices of the network uniformly in $[-0.05, 0.05]$ while all biases are initialized to a constant value at 0.0 with the exception of the *forget gate* biases which is initialised at 1.0 as suggested by Jozefowicz et al. (2015). We also apply 50% dropout (Srivastava et al., 2014) to the non-recurrent connections and clip the norm of the gradients, normalized by the mini-batch size of 32, at 5.0. We also tie the weight matrix used for the transformation in the softmax layer to be the embedding matrix as in Press and Wolf (2016). Thus, the dimensionality of the embeddings is set to 650.

5.2 wikitext2 Setup

For the wikitext2 experiments we trained a multilayer LSTM-RNN with 2 layers of 1000 units. We also used SGD to minimise the average negative log probability of the target words with an initial learning rate of 1.0. We decayed the the learning rate by a factor of 1.15 at each epoch after 14 epochs and we used an early stop counter of 10 epochs. Similarly to the PTB experiment, we initialize the weight matrices of the network uniformly in $[-0.05, 0.05]$ while all biases are initialized to a constant value at 0.0 with the exception of the *forget gate* biases which is initialised at 1.0. For this model we apply 65% dropout to the non-recurrent connections and clip the norm of the gradients, normalized by the mini-batch size of 32, at 5.0. Once again, we tie the weight matrix used for the transformation in the softmax layer to be the embedding matrix. Thus, the dimensionality of the embeddings is set to 1,000.

[2]In other words, the index of the memory starts at timestep 0 instead of timestep 1. Thus, the memory at any given timestep t will be of length $t + 1$.

5.3 Data Manipulation and Batch Processing

When training each model, we use all sentences in the respective training set, but we truncate all sentences longer than 35 words and pad all sentences shorter than 35 words with a special symbol so all have the same length. We use a vocabulary size of 10k for the PTB and 33,278 for the wikitext2. Each of the mini-batches we use for training are then composed of 32 of these sentences taken from the dataset in sequence.

Contrary to the recent trend in the field, we do not allow successive mini-batches to sequentially traverse the dataset. We reinitialize the hidden state of the LSTM-RNN at the beginning of each mini-batch, by setting it to all zeros. Our motivation for not sequentially traversing the dataset is that although sequentially traversing has the advantage of allowing the batches to be processed more efficiently, some dependencies between words may not be learned if batch traversing is in use as the mini-batch boundaries can split sentences. We also found that allowing the initial state of all zeros to be included in the memory when averaging improves the performance of the Average RNN-LM.

5.4 Results

Table 1 presents the results in terms of perplexity of the models trained over the PTB dataset. As we can see, the results obtained by the Averaging RNN-LM are similar to those obtained by the *Attentive* RNN-LMs of Salton et al. (2017). Despite the simple method to retrieve information from the previous timesteps, the Averaging RNN-LM achieves the same level of performance of more complex models with less computation overhead.

Table 2 presents the results in terms of perplexity of the models trained over the wikitext2 dataset. Although the Averaging RNN-LM is still behind the *Attentive* RNN-LMs and the Neural cache model of Grave et al. (2017) on this dataset, the results are encouraging given the simplicity of the Averaging RNN-LM.

However, we should note that none of these models perform at the same level of the state-of-the-art models such as those of Merity et al. (2017) and Takase et al. (2018) as we can see in Tables 1 and 2. These models use advanced regularization techniques and matrix factorization for training the RNN-LMs whilst our Averaging RNN-LM use standard LSTM trainig regime and regularization techniques. Nevertheless, we believe that by adding the regularization scheme of the AWD-LSTM and the direct output connection of AWD-LSTM-DOC to our models we can bridge that performance gap.

6 Discussion

The Averaging LSTM-LM achieves the lowest perplexity for a single model on the PTB (see Table 1). Given the similarity of the results between the *Attentive* RNN-LMs of Salton et al. (2017) and the Averaging LSTM-LM it would appear that our hypothesis that the *Attentive* RNN-LMs was (indirectly) learning to use the dynamics of the LSTM gating mechanism is correct.

Focusing on the results for the wikitext2 dataset, the Neural cache model (Grave et al., 2017) has a higher performance than our model on this dataset. We are not able to estimate the number of parameters for the Neural cache model so we have not included the parameter size of that model in the table. In discussing the wikietext2 results it is worth noting that the *Attentive* RNN-LMs of Salton et al. (2017) and the Averaging LSTM-LM are the only models in Table 2 that reset their memory at each sentence boundary whereas the memory buffers of other models were allowed to span sentence boundaries.

The results for the wikitext2 dataset highlights an interesting trade-off and design choice for memory augmented LSTM-LMs. One approach is to use a dynamic length memory buffer which resets at sentence boundaries and uses a simple mechanism, such as averaging, to construct a representation of the memory to inform the prediction at each timestep. This is the approach we have proposed in this paper. This approach has the advantages of simplicity and that the memory length can be anchored to landmarks in the history, such as sentence boundaries. This approach is most appropriate for sentence based NLP tasks such as sentence based Machine Translation. There is a question, however, regarding whether this approach will scale to very long sequences (such as documents) as averaging over long-histories may result in all histories appearing similar. We have done some initial experiments where we have permitted the memory buffer to hold longer sequences before being reset and the performance of the Averaging LSTM-LM dipped. The alternative approach is to use a larger memory buffer and a

Model	**Params**	**Valid. Set**	**Test Set**
Single Models			
Neural cache model (size = 500) (Grave et al., 2017)	-	-	72.1
Attentive LM w/ *combined* score function (Salton et al., 2017)	14.5M	72.6	70.7
Attentive LM w/ *single* score function (Salton et al., 2017)	14.5M	71.7	70.1
Averaging RNN-LM	14.1M	71.6	69.9
AWD-LSTM (Merity et al., 2017)	24M	60.0	57.3
AWD-LSTM-DOC (Takase et al., 2018)	23M	54.12	**52.38**

Table 1: Perplexity results over the PTB. Please note that we could not calculate the number of parameters for some models given missing information in the original publications.

Model	**Params**	**Valid. Set**	**Test Set**
Averaging RNN-LM	50M	74.6	71.3
Attentive LM w/ *combined* score function (Salton et al., 2017)	51M	74.3	70.8
Attentive LM w/ *single* score function (Salton et al., 2017)	51M	73.7	69.7
Neural cache model (size = 2000) (Grave et al., 2017)	-	-	68.9
AWD-LSTM (Merity et al., 2017)	33M	68.6	65.8
AWD-LSTM-DOC (Takase et al., 2018)	37M	60.29	**58.03**

Table 2: Perplexity results over the wikitext2. Please note that we could not calculate the number of parameters for some models given missing information in the original publications.

more sophisticated retrieval mechanism, for example the Neural cache model of Grave et al. (2017). As the wikitext2 results demonstrate this second approach works well for large datasets where the sentences are in sequence, the cost of this approach being a more complex architecture.

7 Conclusions

In this paper we have highlighted the power of the LSTM gating mechanism and argued that the persistence dynamics of this mechanism can provide useful clues regarding what information is important within a sequence for language modelling. We believe that attending to the information that an LSTM gating mechanism has decided is important in an input sequence at a given timestep (and hence has persisted to a later timestep) is a natural way of deciding what information will be useful again at a subsequent timestep. Even if the information contained in the LSTM is replaced or altered later in the process, we argue that it is relevant to the entire history in proportion to the amount of timesteps it was held. Informed by this hypothesis, in our work we demonstrated that a simple average of the previous LSTM hidden states in memory is an effective mechanism for providing information to the current timestep about previous inputs.

Admittedly, rating the importance of information in terms of the number of timesteps the LSTM persisted it for is a relatively simplistic view of the dynamics of LSTM units and of the complexity of language. Furthermore, implementing this strategy using an average of past states is also a relatively blunt way of instantiating this approach. However, as our results demonstrate this simple approach is effective and we understand this is a starting point. By drawing attention to the signals implicit in the dynamics of LSTM units we hope to contribute to the development of more efficient LMs. At the same time, the fact that the internal dynamics of an LSTM unit may be used to explicitly signal what is important and what should be retrieved from a memory buffer may suggest alternative constraints and opportunities that should be considered in the design of neural units and by doing so contribute to the development of a new class of units for use in RNN-LMs.

Acknowledgments

This research was partly supported by the ADAPT Research Centre, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Funds.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*. volume abs/1409.0473v6.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-Recurrent Neural Networks. *International Conference on Learning Representations (ICLR 2017)* .

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 551–561.

Michal Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly Short Attention Spans in Neural Language Modeling. *5th International Conference on Learning Representations (ICLR'2017)* .

Yarin Gal and Zoubin Ghahramani. 2015. A theoretically grounded application of dropout in recurrent neural networks.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural Comput.* 12(10):2451–2471. https://doi.org/10.1162/089976600300015015.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2003. Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.* 3:115–143. https://doi.org/10.1162/153244303768966139.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. *5th International Conference on Learning Representations (ICLR'2017)* .

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15, pages 2342–2350.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv* abs/1506.02078. http://arxiv.org/abs/1506.02078.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1412–1421.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*. pages 114–119.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *5th International Conference on Learning Representations (ICLR'2017)* .

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. pages 1045–1048.

W. James Murdoch and Arthur Szlam. 2017. Automatic rule extraction from long short term memory networks. *arXiv* abs/1702.02540. http://arxiv.org/abs/1702.02540.

Jared Ostmeyer and Lindsay Cowell. 2017. Machine learning on sequential data using a recurrent weighted average. *arXiv* abs/1703.01253. http://arxiv.org/abs/1703.01253.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. volume abs/1608.05859.

Giancarlo D. Salton, Robert J. Ross, and John D. Kelleher. 2017. Attentive language models. In *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a gpu for statistical machine translation. 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*. pages 11–19.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M. Rush. 2016. Visual analysis of hidden state dynamics in recurrent neural networks. *arXiv* abs/1606.07461. http://arxiv.org/abs/1606.07461.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 4599–4609. https://doi.org/10.18653/v1/D18-1489.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory network for language modeling. *arXiv* abs/1601.01272.