Restricted Recurrent Neural Tensor Networks: Exploiting Word Frequency and Compositionality

Alexandre Salle[1] Aline Villavicencio[1,2]

[1]Institute of Informatics, Federal University of Rio Grande do Sul (Brazil)

[2]School of Computer Science and Electronic Engineering, University of Essex (UK)

alex@alexsalle.com avillavicencio@inf.ufrgs.br

Summary

Increasing the capacity of recurrent neural networks (RNN) usually involves augmenting the size of the hidden layer, with significant increase of computational cost. Recurrent neural tensor networks (RNTN) increase capacity using distinct hidden layer weights for each word, but with greater costs in memory usage. In this paper, **we introduce restricted recurrent neural tensor networks (r-RNTN) which reserve distinct hidden layer weights for frequent vocabulary words while sharing a single set of weights for infrequent words.** Perplexity evaluations show that for fixed hidden layer sizes, r-RNTNs improve language model performance over RNNs using only a small fraction of the parameters of unrestricted RNTNs. These results hold for r-RNTNs using Gated Recurrent Units and Long Short-Term Memory.

Motivation

Sutskever et al. (2011) increased the performance of a character-level language model with a multiplicative RNN (m-RNN), the factored approximation of **a recurrent neural tensor network (RNTN), which maps each symbol to separate hidden layer weights (recurrence matrices)**. Besides **increasing model capacity while keeping computation constant**, this approach has another motivation: viewing the RNN's hidden state as being transformed by each new symbol in the sequence, it is intuitive that **different symbols will transform the network's hidden state in different ways.**

Unfortunately, having **separate recurrence matrices for each symbol requires memory that is linear in the symbol vocabulary size ($|V|$)**. This is not an issue for character-level models, which have small vocabularies, but is **prohibitive for word-level models** which can have vocabulary size in the millions if we consider surface forms.

r-RNTN

To balance expressiveness and computational cost, we propose restricting the size of the recurrence tensor in the RNTN such that memory does not grow linearly with vocabulary size, while still keeping dedicated matrix representations for a subset of words in the vocabulary. We call these Restricted Recurrent Neural Tensor Networks (r-RNTN):

$$h_t = \sigma(W_h x_t + U_h^{f(i(x_t))} h_{t-1} + b_h^{f(i(x_t))})$$

where U_h is a tensor of $K < |V|$ matrices of size $H \times H$, b_h is a $H \times K$ bias matrix with columns indexed by f. The function $f(w)$ maps each vocabulary word to an integer between 1 and K.

We use the following definition for f:

$$f(w) = min(rank(w), K)$$

where $rank(w)$ is the rank of word w when the vocabulary is sorted by decreasing order of unigram frequency.

r-GRU and r-LSTM

GRU:

$$\begin{aligned} r_t &= \sigma(W_h^r x_t + U_h^r h_{t-1} + b_h^r) \\ z_t &= \sigma(W_h^z x_t + U_h^z h_{t-1} + b_h^z) \\ \tilde{h}_t &= \tanh(W_h^h x_t + U_h^h (r_t \odot h_{t-1}) + b_h^h) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \end{aligned}$$

LSTM:

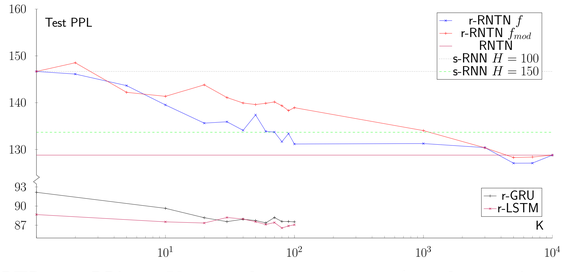
$$\begin{aligned} f_t &= \sigma(W_h^f x_t + U_h^f h_{t-1} + b_h^f) \\ i_t &= \sigma(W_h^i x_t + U_h^i h_{t-1} + b_h^i) \\ o_t &= \sigma(W_h^o x_t + U_h^o h_{t-1} + b_h^o) \\ \tilde{c}_t &= \tanh(W_h^c x_t + U_h^c h_{t-1} + b_h^c) \\ c_t &= i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \\ h_t &= o_t \odot tanh(c_t) \end{aligned}$$

We create r-RNTN GRUs (r-GRU) by making U_h^h and b_h^h input-specific. For r-RNTN LSTMs (r-LSTM), we do the same for U_h^c and b_h^c.

Method	H	PTB # Params	PTB Test PPL	text8 # Params	text8 Test PPL
s-RNN	100	2M	146.7	7.6M	236.4
r-RNTN f	100	3M	131.2	11.4M	190.1
RNTN	100	103M	128.8	388M	-
m-RNN	100	3M	164.2	11.4M	895.0
s-RNN	150	3M	133.7	11.4M	207.9
r-RNTN f	150	5.3M	**126.4**	19.8M	**171.7**

Method	H	PTB # Params	PTB Test PPL
GRU	244	9.6M	92.2
GRU	650	15.5M	90.3
r-GRU f	244	15.5M	**87.5**
LSTM	254	10M	88.8
LSTM	650	16.4M	**84.6**
r-LSTM f	254	16.4M	87.1

Comparison of validation and test set perplexity for r-RNTNs with f mapping ($K = 100$ for PTB, $K = 376$ for text8) versus s-RNNs and m-RNN. r-RNTNs with the same H as corresponding s-RNNs significantly increase model capacity and performance with no computational cost. The RNTN was not run on text8 due to the number of parameters required.



PTB test PPL as K varies from 1 to 10000 (100 for gated networks). At $K = 100$, the r-RNTN with f mapping already closely approximates the much bigger RNTN, with little gain for bigger K, showing that dedicated matrices should be reserved for frequent words.

- As model capacity grows with K, test set perplexity drops.
- Rank-based f mapping more effective than pseudo-random f_{mod} mapping.
- For fixed hidden layer sizes, r-RNTNs yield significant improvements to s-RNNs, GRUs, and LSTMs.
- Given same # of parameters, r-RNTNs outperform s-RNNs, r-GRUs outperform GRUs, using smaller hidden layers.

Conclusion

In this paper, we proposed restricted recurrent neural tensor networks, a model that restricts the size of recurrent neural tensor networks by mapping frequent words to distinct matrices and infrequent words to shared matrices. r-RNTNs were motivated by the need to **increase RNN model capacity without increasing computational costs**, while also satisfying the idea that **some words are better modeled by matrices rather than vectors** (Baroni and Zamparelli, 2010; Socher et al., 2012). We achieved both goals by pruning the size of the recurrent neural tensor network via **sensible word-to-matrix mapping**. Results validated our hypothesis that frequent words benefit from richer, dedicated modeling as reflected in large perplexity improvements for low values of K.

Future Work

Interestingly, **results for s-RNNs and r-GRUs suggest that given the *same number of parameters*, it is possible to obtain higher performance by increasing K and reducing H.** This is not the case with r-LSTMs, perhaps to due to our choice of **which of the recurrence matrices to make input-specific**. We will further investigate both of these phenomena in future work, experimenting with different combinations of word-specific matrices for r-GRUs and r-LSTMs (rather than only U_h^h and U_h^c), and combining our method with recent improvements to gated networks in language modeling (Jozefowicz et al., 2016; Merity et al., 2018; Melis et al., 2018) which we believe are orthogonal and hopefully complementary to our own.

Finally, we plan to compare frequency-based and additional, **linguistically motivated f mappings (for example different inflections of a verb sharing a single matrix)** with mappings learned via conditional computing to measure how external linguistic knowledge contrasts with knowledge automatically inferred from training data.