On Finer Control of Information Flow in LSTMs

Hang Gao(✉) and Tim Oates(✉)

Computer Science and Electrical Engineering Department, University of Maryland,
Baltimore County, 1000 Hilltop Cir, Baltimore, MD 21250, USA
hanggao1@umbc.edu, oates@cs.umbc.edu

Abstract. Since its inception in 1995, the Long Short-Term Memory (LSTM) architecture for recurrent neural networks has shown promising performance, sometimes state-of-art, for various tasks. Aiming at achieving constant error flow through hidden units, LSTM introduces a complex unit called a memory cell, in which gates are adopted to control the exposure/isolation of information flowing in, out and back to itself. Despite its widely acknowledged success, in this paper, we propose a hypothesis that LSTMs may suffer from an implicit functional binding of information exposure/isolation for the output and candidate computation, i.e., the output gate at time $t-1$ is not only in charge of the information flowing out of a cell as the response to the external environment, but also controls the information flowing back to the cell for the candidate computation, which is often the only source of nonlinear combination of input at time t and previous cell state at time $t-1$ for cell memory updates. We propose Untied Long Short Term Memory (ULSTM) as a solution to the above problem. We test our model on various tasks, including semantic relatedness prediction, language modeling and sentiment classification. Experimental results indicate that our proposed model is capable to at least partially solve the problem and outperform LSTM for all these tasks. Code related to this paper is available at: https://github.com/HangGao/ULSTM.git.

Keywords: LSTM · Recurrent neural network · Sequence modeling

1 Introduction

Since its inception in 1995, recurrent neural networks with Long Short Term Memory (LSTM) [1] have shown promising performance on modeling sequential data. Aiming at achieving constant error flow through hidden units, LSTMs are proven to be a scalable method that is both general and effective at capturing long-term temporal dependencies. In fact, LSTMs are widely adopted to advance the state-of-art for many difficult problems in various areas, including handwriting recognition [2,3] and generation [4], language modeling [5–8] and translation [9], image caption generation [10,11], question answering [12], video to text [13] and so on.

M. Berlingerio et al. (Eds.): ECML PKDD 2018, LNAI 11051, pp. 527–540, 2019.
https://doi.org/10.1007/978-3-030-10925-7_32

The key idea behind LSTMs is a complex unit called a memory cell, self-connected and capable of maintaining its state over time, and a set of nonlinear gating units aiming at regulating the information flowing in, out and back to the memory cell. At each time step, as vanilla RNNs, LSTMs are expected to receive a new input, compose it with previous cell state, and then update the cell memory with the guidance of those gates. For standard LSTMs, the fusion of the new input and previous cell state is often mathematically computed as their linear combination followed by a nonlinear transformation (activation). For convenience, we name this fusion as cell input, while referring to the new input as network input. Notice that the cell input is the only source of nonlinear combination involving both network input and cell state, functionally as a candidate for cell update.

However, the above architecture implicitly introduces a bias, that is, the exposure/isolation of information to the external environment and to the generation of the cell input remains the same and can be controlled by the same gate (output gate), i.e., they are functionally tied. This is a strong assumption since the output gate calculated at time $t-1$ is mathematically independent of the new network input coming at time t, but is expected to guide the information flowing out of and back to the cell to generate the cell input in order to update its memory.

In this paper, we propose Untied Long Short Term Memory (ULSTM) as a solution to the above problem. Our idea is to introduce a new type of gate called a retrieve gate, dependent on the network input at each time step, to replace the output gate in the procedure of cell input generation. We only apply the idea to standard LSTMs in our paper, but it can also be generalized to many LSTM variants, e.g., Convolutional LSTM [14], Dynamic Cortex Memory [15] and Group LSTM [16].

We evaluate the proposed model on various tasks, including semantic relatedness prediction, language modeling and sentiment classification. Our experiment results indicate that the proposed model can outperform standard LSTMs in various conditions and is capable to at least partially solve the problem mentioned above.

2 Long Short Term Memory

Initially Long Short Term Memory (LSTM) proposed by [1] included only memory cells, and input and output gates. Targeting at the goal of constant error flow, LSTMs were carefully designed to protect memory cells from perturbation by irrelevant inputs with input gates, and prevent other units from perturbation by currently irrelevant cell content with output gates. Later, forget gates were introduced by [17] to enable LSTMs to reset their own states instead of growing without bound. In general, the transitions of a standard LSTM are defined as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$

$$\tilde{c}_t = n(W_n x_t + U_n h_{t-1} + b_n) \tag{4}$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{5}$$

$$h_t = o_t \odot m(c_t) \tag{6}$$

where Ws and Us are weight matrices, bs are bias vectors, x_t is the network input, i_t, f_t and o_t are the input, forget and output gate respectively, $\tilde{c}_t$ is the cell input, c_t/c_{t-1} and h_t/h_{t-1} are corresponding cell state and output state at the current and previous time steps, and m and n are activation functions, often taken as tanh.

2.1 Peephole Connection

Peephole connections were proposed in [18] to allow all gates to inspect current cell state even when output gates are closed. The transitions of a LSTM with peephole connections are:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + P_i c_{t-1} + b_i) \tag{7}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + P_i c_{t-1} + b_f) \tag{8}$$

$$\tilde{c}_t = n(W_n x_t + U_n h_{t-1} + b_n) \tag{9}$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{10}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + P_o c_t + b_o) \tag{11}$$

$$h_t = o_t \odot m(c_t) \tag{12}$$

where P_i, P_f and P_o are peephole matrices. In practice, we often put constraints on these weight matrices so that they are diagonal, i.e., each gate unit only receives the connection from its own cell. The architecture of a LSTM with peephole connections is presented in Fig. 1.

2.2 Full Gate Recurrence

Mentioned in [19], there is a version of LSTM called Full Gate Recurrence LSTM (FGR-LSTM). The idea is to add connections among all gates. The transitions thus become (with peephole),

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + P_i c_{t-1} + b_i + R_{ii} i_{t-1} + R_{if} f_{t-1} + R_{io} o_{t-1}) \tag{13}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + P_i c_{t-1} + b_f + R_{fi} i_{t-1} + R_{ff} f_{t-1} + R_{fo} o_{t-1}) \tag{14}$$

$$\tilde{c}_t = n(W_n x_t + U_n h_{t-1} + b_n) \tag{15}$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{16}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + P_o c_t + b_o + R_{oi} i_{t-1} + R_{of} f_{t-1} + R_{oo} o_{t-1}) \tag{17}$$

$$h_t = o_t \odot m(c_t) \tag{18}$$

FGR requires nine additional weight matrices, which significantly increases the number of parameters.

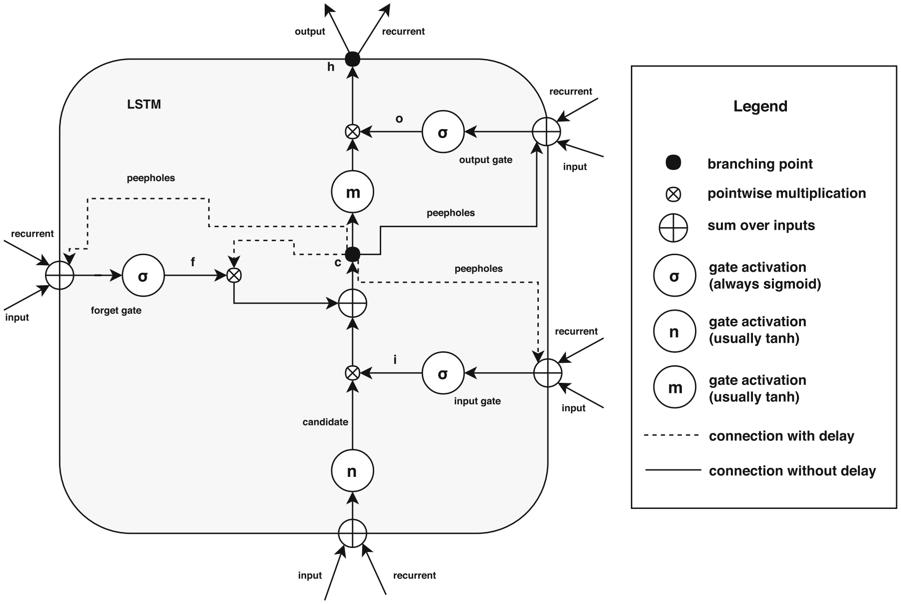

Fig. 1. The architecture of a LSTM with peephole connections

3 Untied Long Short Term Memory

Here we provide the details of our proposed Untied Long Short Term Memory (ULSTM) mentioned above. Although our model can be generalized to many LSTM variants, in this paper, we only focus on the standard LSTM and leave the other variants as future work.

3.1 The Problem

Let us first look at the architecture of a standard LSTM. The information flowing in, back and out of each memory cell is controlled by their corresponding input, forget and output gate. In an alternative view, input gates control the memory write access (W), forget gates manage the memory erase access (E) and output gates determine the memory read access (R). However, if we take a deeper analysis by viewing h_{t-1} as $o_{t-1} \odot \tanh(c_{t-1})$, a summary in Table 1 can be generated.

From that table, one can find that both h_t and c_t have controllers (i_t, f_t or o_t) dependent on current network input x_t, while $\tilde{c}_t$ and all gates are regulated by controllers (o_{t-1}) independent of x_t. The later may lead to a problem we call false exposure/isolation. That is, o_{t-1} incorrectly exposes or isolates information from c_{t-1} in the procedure of computation of $\tilde{c}_t$, i_t, f_t and o_t.

In fact, unlike i_t and f_t, o_t implicitly provides two functions: (1) respond to the external environment at time t; (2) determine the information exposed to

Table 1. Memory cell access at time t

Unit	Direct dependents	Type	Memory accessed	Controller	Controller dependents
$\tilde{c}_t$	x_t, h_{t-1}	R	c_{t-1}	o_{t-1}	x_{t-1}, h_{t-2}
c_t	$\tilde{c}_t$, c_{t-1}	E/W	c_{t-1}	i_t, f_t	x_t, h_{t-1}
h_t	c_t	R	c_t	o_t	x_t, h_{t-1}
i_t	x_t, h_{t-1}	R	c_{t-1}	o_{t-1}	x_{t-1}, h_{t-2}
f_t	x_t, h_{t-1}	R	c_{t-1}	o_{t-1}	x_{t-1}, h_{t-2}
o_t	x_t, h_{t-1}	R	c_{t-1}	o_{t-1}	x_{t-1}, h_{t-2}

the computation of units within the LSTM cell architecture itself at time $t+1$. o_t in a standard LSTM is more likely to behave incorrectly for the latter since it is independent of x_{t+1}, which may be crucial in the decision on whether the information in the memory cell c_t is valuable or not.

Note that LSTMs with peephole connections already provide a solution for gate computation by introducing fixed connections from cells to gate units. In this paper, we focus on proposing a solution for the cell input $\tilde{c}_t$ computation.

3.2 Is Peephole Connection a Solution?

It is almost natural to consider peephole connections for a solution, as has been done with gates. However, this will probably not work. This is due to the fundamental difference between the functions of the cell input and gates. The latter function as controllers on the access of memory cells while the former, instead, function as the candidate to update those cells. A full inspection of the cell state does not logically prevent gates from closing/opening the path from which the information can flow through a cell, but it logically means the information stored in the cell is fully leaked to the candidate (cell input) computation. To prevent cells from being polluted or perturbed during the update procedure, one can only expect the input gates to be capable to not only determine how much to update the cells, but also separate the leaked information from useful one, which is perhaps an even harder problem.

3.3 Retrieve Gates

Since the problem originates from the implicit binding of the two functions of output gates, it is better to find a solution that focuses on detaching them. Out of many potential alternatives, adding a new set of gates to replace output gates and take over their second function is probably the simplest one. We name these gates as retrieve gates and represent them by z in the following. A LSTM with retrieve gates has exactly the same transitions of Eqs. (1)–(3) and (5), (6) as the standard one, with only the modification to Eq. (4),

$$\tilde{c}_t = n(W_n x_t + U_n(z_t \odot \tanh(c_{t-1})) + b_n) \tag{19}$$

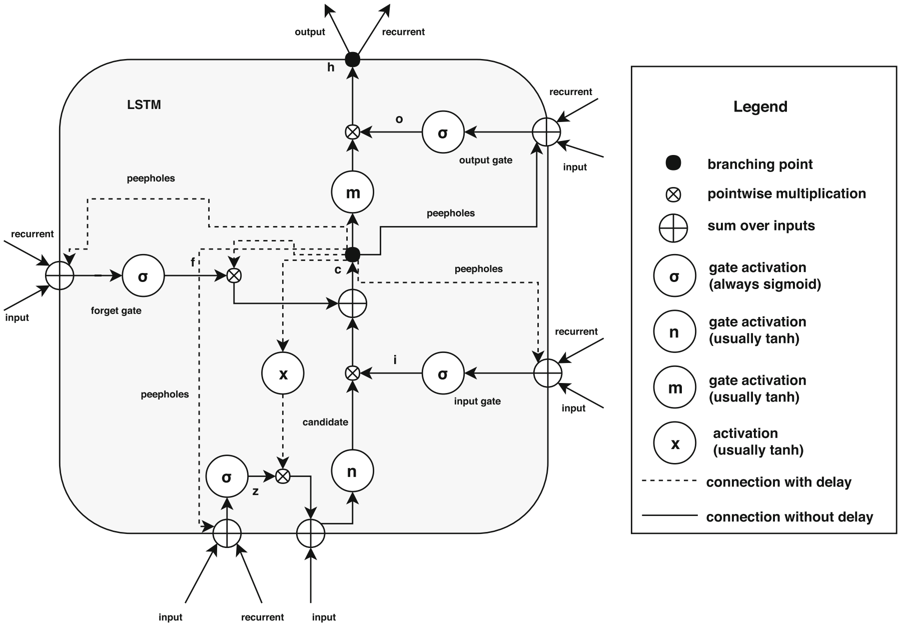

Fig. 2. The architecture of a ULSTM with peephole connections

where,

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{20}$$

Because the solution is inspired by untying the two functions of output gates, we call the model Untied Long Short Term Memory (ULSTM). Note that z_t is now dependent on x_t. We present the architecture of ULSTM in Fig. 2.

4 Experiments

Since our paper focuses on the cell input $\tilde{c}_t$ computation, we only perform tests on ULSTM without peephole connections. We leave the peephole version as future work. In addition, we only compare our model to standard LSTMs and LSTMs with only peephole connections in candidate computations (PLSTM in our experiments), which is mentioned as a potential solution in Sect. 3.2, as our purpose is not to introduce a new model aiming at achieving state-of-art performance for specific tasks.

We evaluate our model on three tasks: (1) predicting the semantic relatedness of sentence pairs; (2) word level language modeling and (3) sentiment classification of sentences sampled from movie reviews.

4.1 Semantic Relatedness

For a given pair of sentences, the semantic relatedness task is to predict a human-generated rating of the similarity of the two sentences in meaning. We use the Sentences Involving Compositional Knowledge (SICK) dataset introduced by [20]. Consisting of 9927 sentence pairs, this dataset is pre-split into train/valid/test sets with ratio 4500/500/4927. All sentences are derived from existing image and video description datasets, with each pair annotated with a relatedness score $y \in [1, 5]$. The higher the score, the more related the pair of sentences are. We adopt the same similarity model and objective function as [21], with only a slight difference on the activation function choice for h_s. Instead of,

$$h_s = \sigma(W^{(x)} h_x + W^{(+)} h_+ + b^{(h)}) \tag{21}$$

We use,

$$h_s = ReLU(W^{(x)} h_x + W^{(+)} h_+ + b^{(h)}) \tag{22}$$

Table 2. Evaluation results of LSTM, PLSTM and ULSTM on SICK test data. The best results in each subsection are marked as bold

Model	Hidden size	Parameters	Pearson's γ	Spearman's ρ	MSE
LSTM	150	1009205	0.8545	0.7931	0.2774
PLSTM	150	1009355	0.8542	0.7926	0.2782
ULSTM	150	1077005	**0.8591**	**0.7973**	**0.2686**
LSTM (o=1)	150	<1009205	0.8562	0.7955	0.2747
PLSTM (o=1)	150	<1009355	0.8568	0.7948	0.2737
ULSTM (o=1)	150	<1077005	**0.8604**	**0.7984**	**0.2662**
LSTM	180	1088045	0.8531	0.7896	0.2803
PLSTM	180	1088225	0.8518	0.7900	0.2811
ULSTM	180	1174805	**0.8609**	**0.8020**	**0.2674**
LSTM (o=1)	180	<1088045	0.8550	0.7932	0.2747
PLSTM (o=1)	180	<1088225	0.8566	0.7956	0.2738
ULSTM (o=1)	180	<1174805	**0.8603**	**0.8022**	**0.2674**
LSTM	210	1174085	0.8525	0.7916	0.2794
PLSTM	210	1174295	0.8516	0.7905	0.2825
ULSTM	210	1281605	**0.8630**	**0.8039**	**0.2641**
LSTM (o=1)	210	<1174085	0.8534	0.7906	0.2789
PLSTM (o=1)	210	<1174295	0.8532	0.7936	0.2793
ULSTM (o=1)	210	<1281605	**0.8620**	**0.8028**	**0.2657**
LSTM	245	1283565	0.8476	0.7842	0.2902
PLSTM	245	1283810	0.8490	0.7890	0.2875
ULSTM	245	1417580	**0.8609**	**0.8024**	**0.2664**
LSTM (o=1)	245	<1283565	0.8521	0.7907	0.2816
PLSTM (o=1)	245	<1283810	0.8509	0.7908	0.2845
ULSTM (o=1)	245	<1417580	**0.8537**	**0.7937**	**0.2776**

We adopt the publicly available Glove vectors [22] as the initialization of word embeddings. Following [21], we do not fine tune these embeddings during the training procedure. For the optimization algorithm, we choose Adagrad [23], with a learning rate of 0.03 and a weight decay rate of 0.0001. We set the batch size to be 25 and number of epochs to be 10. For the similarity model, the hidden layer size is set to be 50. We use Pearson's γ, Spearman's ρ and MSE as the evaluation metrics. We adopt early stopping and perform prediction on the test data with the model of the highest Pearson's γ on the validation data. For each set of hyper-parameters, we report the mean of 5 independent runs with random seeds 1234/2341/3451/3651/3851.

For this task, we seek to compare ULSTM, PLSTM and LSTM with (1) varying hidden size; (2) output gates manually fixed to 1 or not; (3) the same number of parameters. We list all evaluation results in Table 2.

Varying Hidden Size. When we look at the results of LSTM, PLSTM and ULSTM with varying hidden size from 150 to 245, we find that ULSTM consistently outperforms LSTM and PLSTM, regardless of whether output gates are fixed to 1 or not. Besides, ULSTM can benefit from increased hidden size until a certain point, while the performance of LSTM keeps dropping as the hidden size increases. PLSTM, on the other hand, shows a similar but weaker trend as LSTM. Since in this task error signals for parameter tuning only come from the end of each sentence, ULSTM seems to be better at exploiting the benefits brought by increased storage capacity, while LSTM and PLSTM, on the contrary, suffer from it due to the lack of error signals to correct memory access management, as the former has lower requirements on the behavior of its output gates or input gates.

Fixed Output Gates or Not. We take another look at the performance of each model with/without fixing output gates. Out of 4 different hidden sizes, when the output gates are not fixed to 1, LSTM noticeably performs worse under 3 cases and PLSTM performs worse under all cases, while for ULSTM, in most scenarios, the performance is roughly the same or even better than the one with fixed output gates. A possible explanation is that for ULSTM, output gates are only expected to affect information exposure to the external environment or gate computation. As discussed above, gates can be correctly opened/closed even with information wrongfully exposed/isolated. So as long as the external environment (the similarity model) is robust, the performance of ULSTM is expected to remain stable or slightly worse. This is not the case for LSTM, in which the output gates play an important role in memory candidate generation or for PLSTM, where information in cells is simply always fully exposed to the generation procedure.

The Same Number of Parameters. By comparing LSTM, PLSTM and ULSTM with different hidden sizes but roughly the same number of parameters, one can conclude that ULSTM, for this task, always performs better than either PLSTM or LSTM, if given similar number of parameters.

4.2 Language Modeling

The goal of a language model is to compute the probability of a sentence or a sequence of words. We use a preprocessed version of the Penn Treebank data set (PTB) introduced in [24], which is also adopted by [5]. PTB has long been a central dataset for evaluation of language models. The data set is preprocessed so that it does not contain capital letters, numbers or punctuation. The vocabulary includes around 10000 unique words, which is small compared to many modern datasets.

Table 3. Valid and Test PPL on PTB of LSTM, PLSTM and ULSTM with varying number of layers. The best results in each subsection are marked as bold

Model	Hidden size	Layers	Parameters	Valid PPL	Test PPL
AWD-ULSTM	400	1	5614000	**84.23**	**81.76**
AWD-PLSTM	400	1	5293600	85.62	83.14
AWD-LSTM	400	1	5293200	86.17	83.73
AWD-ULSTM	400	2	7218000	**70.05**	**67.38**
AWD-PLSTM	400	2	6577200	71.16	68.64
AWD-LSTM	400	2	6576400	72.83	70.18
AWD-ULSTM	400	3	8822000	**68.47**	**65.41**
AWD-PLSTM	400	3	7860800	69.45	67.06
AWD-LSTM	400	3	7859600	69.87	67.03
AWD-ULSTM	400	4	10426000	**69.58**	**66.42**
AWD-PLSTM	400	4	9144400	70.01	67.23
AWD-LSTM	400	4	9142800	70.58	67.51
AWD-ULSTM	400	5	12030000	**70.12**	**67.05**
AWD-PLSTM	400	5	10428000	71.1	68.51
AWD-LSTM	400	5	10426000	71.47	68.78

We adopt an implementation of AWD-LSTM [5], which regulates LSTMs with various techniques that do not make any modification to existing model architectures, such as variable length sequence, DropConnect and so on. For this task, we randomly initialize word embeddings with a uniform distribution of $(-0.1, 0.1)$ with dimension of 400. The maximum sequence length is set to be 70. We apply dropout with rate 0.5 on the decoder, rate 0.4 on embeddings, and rate 0.25 between hidden layers of RNNs. In addition, words from the embedding layer are dropped with rate 0.1 and RNN hidden to hidden weight connections are randomly dropped with rate 0.5. We set the random seed to be 141, L2 regularization rate on RNN activation to be 2, slowness regularization rate on RNN activation to be 1 and weight decay rate to be $1.2e^{-6}$. For the optimization algorithm, we adopt SGD with learning rate of 30 and switch to NT-ASGD [5]

Table 4. Valid and Test PPL on PTB of LSTM, PLSTM and ULSTM with varying hidden size. The best results in each subsection are marked as bold

Model	Hidden size	Layers	Parameters	Valid PPL	Test PPL
AWD-ULSTM	300	2	6467000	**72.01**	**69.01**
AWD-PLSTM	300	2	5976300	73.49	70.52
AWD-LSTM	300	2	5975600	74.55	71.81
AWD-ULSTM	400	2	7218000	**70.50**	**67.38**
AWD-PLSTM	400	2	6577200	71.16	68.64
AWD-LSTM	400	2	6576400	72.83	70.18
AWD-ULSTM	500	2	8069000	69.65	**67.12**
AWD-PLSTM	500	2	7258100	**69.56**	67.44
AWD-LSTM	500	2	7257200	71.06	68.43
AWD-ULSTM	600	2	9020000	68.83	**65.97**
AWD-PLSTM	600	2	8019000	**68.55**	66.30
AWD-LSTM	600	2	8018000	69.42	67.03
AWD-ULSTM	720	2	10293200	67.53	**64.77**
AWD-PLSTM	720	2	9037680	**67.48**	65.23
AWD-LSTM	720	2	9036560	68.63	66.17
AWD-ULSTM	1000	2	13824000	65.49	62.95
AWD-PLSTM	1000	2	11862600	**64.98**	**62.86**
AWD-LSTM	1000	2	11861200	66.09	63.83
AWD-ULSTM	1170	2	16350200	64.55	**62.14**
AWD-PLSTM	1170	2	13883730	**64.30**	62.16
AWD-LSTM	1170	2	13882160	65.14	62.94
AWD-ULSTM	1350	2	19340000	64.04	61.73
AWD-PLSTM	1350	2	16275750	**63.83**	**61.43**
AWD-LSTM	1350	2	16274000	64.48	62.11
AWD-ULSTM	1500	2	22079000	63.97	61.21
AWD-PLSTM	1500	2	18467100	**62.71**	**60.66**
AWD-LSTM	1500	2	18465200	63.68	61.40

if its trigger criterion is satisfied. We set the logging interval to be 1 and the non-monotone interval to be 5 for NT-ASGD. All gradients are clipped with absolute value 0.25. The batch size is set to be 20 and the maximum number of epochs is 500.

For this task, we seek to compare ULSTM, PLSTM and LSTM with (1) different stacking layers; (2) varying hidden size; (3) the same number of parameters. Note that unlike semantic relatedness prediction, in this task, both models may receive error signals at each time step. We list all evaluation results in Tables 3 and 4.

Different Stacking Layers. The comparison among ULSTM, PLSTM and LSTM with fixed hidden size but different number of layers is presented in Table 3. The results indicate that along with the increased number of layers, ULSTM always outperforms LSTM on both validation and test data, while PLSTM is usually better but occasionally worse than LSTM with a small margin.

Varying Hidden Size. We show the comparison among ULSTM, PLSTM and LSTM with fixed number of layers but varying hidden sizes in Table 4. It is clear that both ULSTM and PLSTM outperform LSTM on test data with various hidden sizes, indicating the existence of the proposed problem for standard LSTMs. However, when the hidden size is not very large ($<$1000), ULSTM outperforms PLSTM while when the hidden size grows large ($\geq$1000), the latter starts to outperform the former. A possible explanation is that the increase of hidden size may reduce the benefits brought by the retrieve gates as cells have more storage capacity for redundant information, thus (1) false isolation is unlikely to happen since it requires all relevant gates to be closed at the same time; (2) false exposure is more likely to occur as it is hard to make sure that not a single relevant gate is open, when the number of cells is large. As a result, as the hidden size increases, the performance gap between ULSTM and LSTM decreases and since it is difficult to prevent false exposure with large hidden size, PLSTM seems to benefit more by simply adding direct peephole connections so that other gates are trained to work with irrelevant noise that is always present. This trend does not occur above because in Sect. 4.1, there is not enough error signal at every time step to correct gate behavior.

The Same Number of Parameters. From Table 4, ULSTM can outperform LSTM with the same number of parameters when the hidden size is still small, but along with the increase of hidden size, LSTM starts to outperform ULSTM, which is probably due to the reasons mentioned above. However, PLSTM usually outperforms LSTM with roughly the same number of parameters, regardless of the number of layers or the hidden sizes.

4.3 Sentiment Classification

In this task, we predict the sentiment of sentences sampled from movie reviews. We use the Stanford Sentiment Treebank [25]. There are two possible subtasks for this dataset, but we only focus on the fine-grained classification task over five classes: very negative, negative, neutral, positive, very positive. We use the train/valid/test split provided by the dataset.

For this task, we do not control variables when comparing ULSTM, PLSTM and LSTM. Instead we perform hyper-parameter search on the number of layers from the set [1, 2, 3] and hidden size from the set [50, 100, 150, 200, 250, 300] to find the best hyper-parameter setting for each model. We initialize word embeddings with Glove vectors [22] and fine-tune them during the training procedure. We use SGD followed by NT-ASGD [5] if the trigger criterion is satisfied. We set the logging interval to be 1 and the non-monotone interval to be 5 for

NT-ASGD. The learning rate is set to be 1. We apply dropout with rate 0.4 on embeddings, rate 0.25 between hidden layers of RNNs. In addition, RNN hidden to hidden weight connections are randomly dropped with rate 0.5. We set the random seed to be 141, L2 regularization rate on RNN activation to be 2, slowness regularization rate on RNN activation to be 1 and weight decay rate to be $1.2e^{-6}$. The number of epochs is set to be 50.

We report the test accuracy of each model with the hyper-parameter set chosen on validation data in Table 5. ULSTM still shows better performance than LSTM for the sentiment classification task, for both validation and test accuracy. Note that both LSTM and ULSTM share the same hyper-parameter set when they achieve the highest validation accuracy, indicating that like other tasks, ULSTM is likely to perform better than LSTM given the same number of layers and hidden size.

However, PLSTM in this experiment performs worse than either LSTM or ULSTM on both validation and test data. Note that similar to Semantic Relatedness task, all models only receive error signals at the end of a sequence from the classifier. It is possible that the lack of error signals makes it difficult for PLSTM to adjust its gate computation to work with noise constantly present, leading to worse results.

Table 5. Results of LSTM, PLSTM and ULSTM on Stanford Sentiment Treebank. The best result is marked as bold

Model	Hidden size	Layers	Valid Acc	Test Acc
LSTM	50	2	48.56	48.28
LSTM	200	2	48.56	47.72
PLSTM	200	1	48.28	46.82
ULSTM	50	2	**49.71**	**48.66**

5 Conclusion and Future Work

In this paper, we address the problem of implicit functional binding of output gates in LSTM, which may lead to false exposure/isolation of information for the cell input computation. We propose a model called Untied Long Short Term Memory (ULSTM) that introduces a new set of gates as a solution. We evaluate our model on three tasks: (1) semantic relatedness prediction; (2) language modeling; (3) sentiment classification. The experimental results indicate: (a) ULSTM consistently outperforms LSTM on all the three tasks, given the same number of layers and hidden size; (b) ULSTM usually outperforms LSTM with the same number of parameters when the hidden size is small, but it is not necessarily true if we increase the hidden size; (c) On the other hand, LSTM may benefit or suffer from large hidden size, depending on the task; (d) Although LSTM with peephole connections in cell input computation (PLSTM) can sometime work better than LSTM, or even ULSTM, it is highly task-dependent and inconsistent.

We seek to generalize our model to large data sets in the future. And it is also possible to apply the idea to other LSTM variants, e.g., Convolutional LSTM. Besides the idea of adding a new set of gates which introduces more parameters, we are also interested in looking for other alternative solutions that keep or reduce the number of parameters, which may be more computationally efficient.

References

1. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
2. Graves, A., et al.: A novel connectionist system for unconstrained handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 855–868 (2009). https://doi.org/10.1109/TPAMI.2008.137
3. Pham, V., et al.: Dropout improves recurrent neural networks for handwriting recognition. In: 2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE (2014). https://doi.org/10.1109/ICFHR.2014.55
4. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
5. Merity, S., Keskar, N.S., Socher, R.: Regularizing and optimizing LSTM language models. arXiv preprint arXiv:1708.02182 (2017)
6. Yang, Z., et al.: Breaking the softmax bottleneck: a high-rank RNN language model. arXiv preprint arXiv:1711.03953 (2017)
7. Inan, H., Khosravi, K., Socher, R.: Tying word vectors and word classifiers: a loss framework for language modeling. arXiv preprint arXiv:1611.01462 (2016)
8. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
9. Luong, M.-T., et al.: Addressing the rare word problem in neural machine translation. arXiv preprint arXiv:1410.8206 (2014)
10. Vinyals, O., et al.: Show and tell: a neural image caption generator. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2015). https://doi.org/10.1109/CVPR.2015.7298935
11. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: International Conference on Machine Learning (2015)
12. Wang, D., Nyberg, E.: A long short-term memory model for answer sentence selection in question answering. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), vol. 2 (2015)
13. Venugopalan, S., et al.: Sequence to sequence-video to text. In: Proceedings of the IEEE International Conference on Computer Vision (2015). https://doi.org/10.1109/ICCV.2015.515
14. Xingjian, S.H.I., et al.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in Neural Information Processing Systems (2015)
15. Otte, S., Liwicki, M., Zell, A.: Dynamic cortex memory: enhancing recurrent neural networks for gradient-based sequence learning. In: Wermter, S., et al. (eds.) ICANN 2014. LNCS, vol. 8681, pp. 1–8. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11179-7_1

16. Kuchaiev, O., Ginsburg, B.: Factorization tricks for LSTM networks. arXiv preprint arXiv:1703.10722 (2017)
17. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM, pp. 850–855 (1999)
18. Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with LSTM recurrent networks. J. Mach. Learn. Res. **3**, 115–143 (2002)
19. Greff, K., et al.: LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2222–2232 (2017). https://doi.org/10.1109/TNNLS.2016.2582924
20. Marelli, M., et al.: Semeval-2014 task 1: evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014) (2014)
21. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075 (2015)
22. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014). https://doi.org/10.3115/v1/D14-1162
23. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**, 2121–2159 (2011)
24. Mikolov, T., et al.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association (2010)
25. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (2013)