An Improved Differential Neural Computer Model Using Multiplicative LSTM

Khushmeet S. Shergill[✉], K. Sharmila Banu, and B. K. Tripathy

School of Computer Science and Engineering, VIT University, Vellore, India
{khushmeetsingh1996,sharmila.k,tripathybk}@vit.ac.in

Abstract. Artificial neural networks excel at doing specific tasks like image recognition, sequence learning, machine translation. But, the direction of research has moved towards the creation of more general purpose neural network architectures. Recently, DeepMind introduced Differentiable Neural Computer (DNC), with an external memory system that is capable of working on complex data structures. DNC can infer from graph problem, solve block puzzle using reinforcement learning and so on. DNC uses LSTM as controller network that manipulates the memory matrix. In this paper, we introduce a change to DNC architecture by replacing LSTM network with multiplicative LSTM and measure the performance of the improved model by training it on three different tasks; namely question answering task using bAbI dataset, character level modelling using harry potter text and planning search using air cargo problem. We compared the performance of the previous model to determine its behavior.

Keywords: LSTM · Memory · Planning search · Recurrent neural networks

1 Introduction

Differentiable Neural Computer (DNC) is a powerful neural network model coupled with external memory matrix. The model itself consists of two main components, controller and memory. Controller is recurrent neural network that acts like the CPU of the model, whereas memory is just an $N \times W$ matrix which can be thought of as the RAM of the model. The operations of the manipulating the memory is learned through gradient descent. The system is end to end differentiable.

Memory consists of weights, which represents the degree to which the particular location is involved in read or write operation. System has read and write heads that are involved in memory manipulation.

Differentiable Neural Computer can be thought of as a LSTM network, in a sense that they both try to remember things. The only difference is that DNC is given an external memory to remember things where as LSTM uses its hidden states for learning long term dependencies. But LSTM fall short in this regard.

DNC has taken inspiration from mammalian hippocampus. For e.g. humans recall the memories in the order they were remembered. Same happens in DNC with temporal link matrix, that retrieves memory in the same sequence they were written. Another is

I. Zelinka et al. (Eds.): ICSCS 2018, CCIS 837, pp. 283–290, 2018.
https://doi.org/10.1007/978-981-13-1936-5_31

memory allocation is DNC, which is similar to dentate gyrus region of the hippocampus that performs neurogenesis.

2 Related Works

Artificial Neural Networks were created with the goal of trying to replicate thinking and remembering, which are the hallmark of human brain. With some setbacks ANNs started to perform well on many tasks, where brain excelled, such as image and speech recognition, machine translation and so on. But these neural networks still weren't good at doing inferencing and answering complex queries. This changed when Recurrent Neural Networks (RNN) were introduced. They have this feedback loop, that let them remember input through their hidden states. RNNs take sequence as vector $\left[x_1, x_2, x_3 \ldots x_n\right]$, $x_t \in \mathbb{R}^X$ and each output is a prediction sequence given as probability distribution with the help of softmax function. The hidden state $h_t \in \mathbb{R}^H$ is updates as a linear combination of input x_t and hidden state in the previous time step h_{t-1}. Later Bengio et al. [1] showed that RNNs suffer from vanishing gradient problem, where the model cannot store long term dependencies and training start to suffer.

In 1997 Schmidhuber and Hochreiter [2] introduced Long Short-Term Memory Network which solved vanishing gradients problem by introducing forget gate and new memory gate, that allowed hidden state to either pass through time (remembered) or not (forgot). This enabled the modelling of sequence with much greater length than before.

This was a significant win over RNNs and are very popular in sequence modelling, but still fall short when it comes to very long input sequence (limit of LSTMs). It can be seen in tasks like question answering.

To solve this, neural networks were given a separate memory component that can remember inputs and recall those inputs at later time steps. There are several networks that have memory component. Dynamic Memory Networks (DNM) from Kumar et al. [3] was built for question answering. It has semantic memory module that had GloVe vectors that are used to create sequence of word embeddings. The input module processes the text into a set of vectors. The episodic memory consists of 2 GRU modules. Outer GRU generates the memory vector from episodes, which are generated by inner GRU by moving over the facts from input modules.

Neural Turing Machine (NTM) (Graves et al.) [4] is a predecessor to DNC. It is different from memory networks in a sense that instead of just working on a specific task like question answering, they are built to perform algorithmic tasks (generalize to perform various tasks). The model is similar to DNC, meaning that it has same architecture as DNC, but differs in access mechanism. NTM uses location-based addressing, restricting how memory is being written (contiguous blocks). NTMs has no way to guarantee non-overlapping of memory locations. NTMs cannot free used memory locations.

3 High Level Overview of DNC

Model works by giving an input to the controller $x_t \in \mathbb{R}^X$ at every time step and it then emits an output vector $y_t \in \mathbb{R}^Y$. Along with input vector x_t controller receives a set of read vectors from the previous time step. Controller then emits an interface vector, that defines the way the controller will interact with the memory matrix denoted formally as $M \in \mathbb{R}^{N \times W}$. Both the input vector and read, write vectors are concatenated to obtain single vector with is then input into the controller.

$$X = \left[x_t, r^1 \ldots r^n\right]$$

After one iteration, model emits and output vector and an interface vector (δ) consisting of parameters for interacting with memory.

Reading memory is done by using read weightings to compute weighted average of the content of location, which are called read vectors, which are then sent to the controller network, therefore giving it access to memory content.

For write operation, a single write weighting $\left(w_t^w\right)$ calculated from memory allocation mechanism, erase vector and write vector (both taken from interface vector) is used to modify memory content.

DNC uses three mechanisms for concentrating on specific memory location while reading and writing. This increases the effectiveness of the model in storing and recalling. DNC uses content bases attention mechanism, which uses similarity between memory vector and the key vector. This is helpful when recalling some fact or finding a similar pattern. Second is memory allocation. DNC reuses memory location by defining usage vector u_t (for complete information refer to original paper by Graves et al.). Third is temporal linkage. This allows the DNC to recall facts in the order they were presented, simulating how human brain recall facts.

4 Multiplicative LSTM as Controller Network

Multiplicative LSTM (mLSTM) (Krause et al.) [5] is a hybrid architecture that combines LSTM and Multiplicative RNN (Sutskever et al.) [6]. Multiplicative RNN modifies Tensor RNN (Weston et al.) [7] to have hidden weight matrix for every input.

$$h_t = \tanh\left(W_{hx}x_t + W_{hh}^{(x_t)}h_{t-1} + b_h\right)$$
$$o_t = W_{oh}h_t + b_o$$

The hidden matrix used for given input is

$$W_{hh}^{(x_t)} = \sum_{n=1}^{N} W_{hh}^{(n)} x_t^{(n)}$$

Where $x_t^{(n)}$ is a one hot encoding of the input, and N is the dimensionality of the input. W_{hh} is a tensor here, which can be seen as a list of hidden matrices. With increasing dimensionality of x_t, W_{hh} becomes immensely huge to work with. Therefore W_{hh} is factorized as follows.

$$W_{hh}^{(x_t)} = W_{hf} \cdot diag\left(W_{fx}x_t\right) \cdot W_{fh}$$

To get mLSTM, hidden state from mRNN is plugged into gating units of LSTM.

$$\begin{aligned} m_t &= \left(W_{fx}x_t\right) \odot \left(W_{fh}h_{t-1}\right) \\ h_t &= W_{hx}x_t + W_{hm}m_t \\ i_t &= \sigma\left(W_{ix}x_t + W_{im}m_t\right) \\ o_t &= \sigma\left(W_{ox}x_t + W_{om}m_t\right) \\ f_t &= \sigma\left(W_{fx}x_t + W_{fm}m_t\right) \end{aligned}$$

5 Experimental Results and Analysis

The first experiment is done on *bAbI dataset* (Weston et al.) [8]. bAbI contains set of tasks and for each task there are 1000 questions for training and 1000 for testing. These tasks are designed to see whether network is able to answer query where deduction is required or can it count or can it answer questions in yes/no style etc. For full details on bAbI task, please refer to the paper by Weston et al.

In preprocessing phase, inputs and targets are encoded as vectors of length 100 by constructing a lexicon of unique words and padded zeros at the beginning if sequence length is small and is serialized to be used later during training phase.

During training, input and output vectors are taken one at a time, since batch size is 1 and are then converted to one hot vectors which are then fed to the model.

Two different DNC models are used DNC1 with LSTM as controller network and DNC2 with mLSTM variant both using 256 cells. Rest of the parameters are same as of DeepMind DNC. Both models are run for 300,000 iterations.

Following table compares DeepMind DNC with our model (Table 1).

Model performed poorly on Path finding and Basic Induction, but performed really well on Agent Motivation, Compound Coreference and Basic Coreference. Model if trained further could lower the error rate on every task. Model is further compared with DeepMind's DNC with their results taken from the paper.

Above Fig. 1 displays loss for 300,000 iterations but is showing only last 200,000 iterations because the training was later continued from a checkpoint. Loss fluctuation is high, but model was able to perform moderately accurate on bAbI tasks.

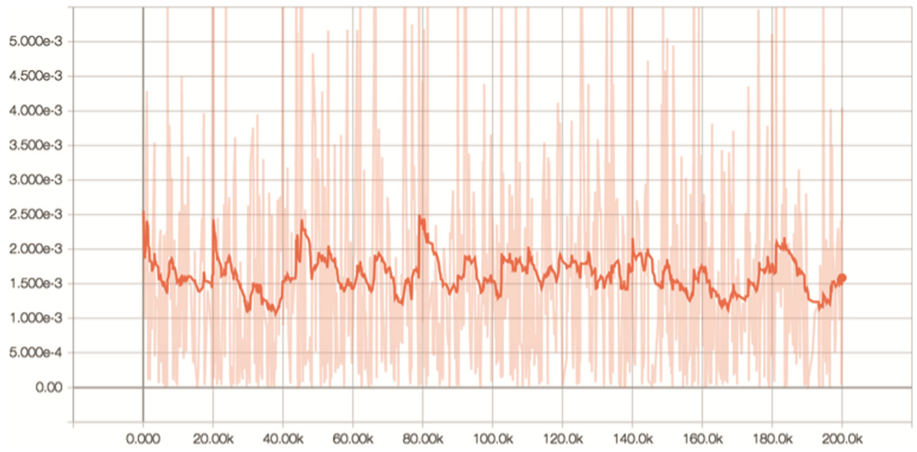

Fig. 1. Loss vs Iterations (bAbI task)

Table 1. Results for bAbI task

Task	DeepMind DNC	DNC
Positional reasoning	24.1	43.75
Two argument relation	0.0	5.69
Agent motivation	0.0	1.10
Counting	0.2	9.00
Single supporting fact	0.0	13.30
Path finding	0.1	90.23
Basic deduction	0.0	53.19
Three supporting fact	2.4	30.70
Indefinite knowledge	0.2	27.04
Three argument relation	0.5	13.20
Negation	0.0	4.40
Conjunction	0.1	8.40
Basic coreference	0.0	4.14
Time reasoning	0.3	27.98
Size reasoning	4.0	8.43
Compound coreference	0.0	1.05
Lists	0.1	7.70
Basic induction	52.4	63.19
Two supporting fact	1.3	30.37
Yes/No	0.0	5.80

The second experiment is *character level language modelling*. This means that DNC tries to model probability distribution of the next character is the given sequence. For the DNC to work, output from the controller is passed through a softmax classifier with number of units equals the length of the vocabulary.

The dataset used is seven harry potter books scraped from internet archive [9]. The input is converted to one hot encoded vectors with dimensions having (100, 100, 83) which is represented as (*batch size*, *sequence steps*, *one hot vector length*). Loss function used is softmax cross entropy loss. Standard DNC configuration is used with 1 write head, 4 read heads and 256 memory locations and multiplicative LSTM as controller network. The model reaches 0.04076 error rate after 26000 iterations. To test the model, character is passed through the model, the output for that corresponding input is used as input for the next time step, therefore running for 2000 iteration. Following is the output for 2000 step using "Magic" as the starting input. Following is an excerpt from the output.

Magic Leady was a few more points into the darkness to the common room they did, and they didn't take their way to the corridor. They could have to be carried the sign that she had taken a sharp polish when he wanted to trust her. He was telling him to get out of the move; he has taken a first year threat on them, the moment she was about to see it too, but they were tenting to have a good stone way, and then he started at him. "It's be there to do what te was stopped, you said to his stomach," said Dumbledore, starting to strain to Harry into her hand. "Well, if you come back there. The Muggle birth was a bulging second time. … They didn't look like it has the bar in the silence and take it."

Observing the above text, it can be seen that individual words generated were meaningful but the sentence structure did not make any sense. Some of the words that is particular to the input text, for example "Dumbledore" or "Muggle" were correctly generated, but later in the sentence, Dumbledore was referred as her, concluding that words as a collection does not possess any meaning. But overall grammar of the text is correct.

The model was able to converge quite nicely at 26,000 iterations and text resembles with the input given. Various words unique to this text, were generated accurately by the model. Loss vs Iteration of model is present in Fig. 2.

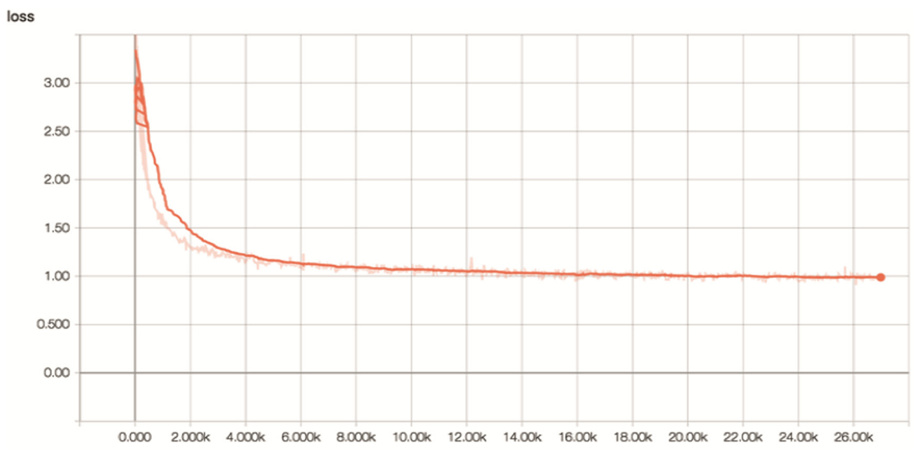

Fig. 2. Loss vs Iteration

The third experiment is *planning problem*. Air cargo planning problem is chosen from [8] Russel and Norvig. Given initial conditions, goal and list of actions that can be performed, model's task is to find optimal number of steps to reach that goal.

For example, given an initial condition and Goal and an action schema (Fig. 3).

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ Cargo(C1) ∧ Cargo(C2)
            ∧ Plane(P1) ∧ Plane(P2)∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
Action(Load(c, p, a), PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
            EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a), PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
            EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to), PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧
Airport(to)
            EFFECT: ¬ At(p, from) ∧ At(p, to))
```

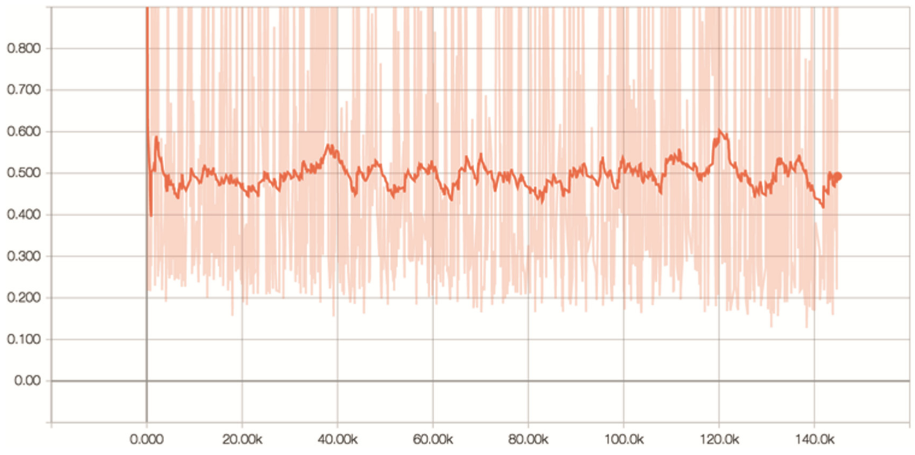

Fig. 3. Loss vs Iterations (Planning problem)

Following this data, the goal will be achieved in 6 steps, by following the plan given below.

Load (C1, P1, SFO)
Load (C2, P2, JFK)
Fly (P1, SFO, JFK)
Fly (P2, JFK, SFO)
Unload (C1, P1, JFK)
Unload (C2, P2, SFO)

DNC's task is to figure out the plan in minimum number of steps.

DNC has to figure out the correct order of the item in the tuple, like the arguments given to the action $(C1, P1, SFO)$ and to determine which action to use, Load or Fly or Unload.

The experiment is conducted with 2 cargos, 2 planes and 2 airports. After training for 2 days for 140,000 iterations on NVidia K80 12 GB GPU, it gave 62.5% accuracy on 8 planning problems. Graph below shows model reaches 0.2 loss.

Model is taking long time to converge, with another 100,000 iterations required to reach loss below 0.1.

6 Conclusion

In this paper we proposed a new DNC model with the LSTM being replaced with multiplicative LSTM. The proposed model has a slightly larger training time in comparison to the previous model. But, being a generalised model it is applicable to larger class of problems. Moreover, the new DNC model tries to emulate algorithmic tasks. When executed in powerful computers with higher processing speed there are endless possibilities for the proposed model.

References

1. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**, 157–166 (1994)
2. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
3. Kumar, A., et al.: Ask me anything: dynamic memory networks for natural language processing. In: International Conference on Machine Learning, pp. 1378–1387, June 2016
4. Graves, A., Wayne, G., Danihelka, I.: Neural turing machines (2014). arXiv preprint arXiv:1410.5401
5. Krause, B., Lu, L., Murray, I., Renals, S.: Multiplicative LSTM for sequence modelling (2016). arXiv preprint arXiv:1609.07959
6. Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 1017–1024 (2011)
7. Weston, J., et al.: Towards AI-complete question answering: a set of prerequisite toy tasks (2015). arXiv preprint
8. Russell, S.J., Norvig, P., Canny, J.F., Malik, J.M., Edwards, D.D.: Artificial Intelligence: A Modern Approach, vol. 2, no. 9. Prentice Hall, Upper Saddle River (2003)
9. Rowling, J.K.: Internet Archive. https://archive.org/details/HarryPotterCompleteCollection