IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 9, 2018, accepted February 22, 2018, date of publication March 6, 2018, date of current version March 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2812766

On the Duration, Addressability, and Capacity of Memory-Augmented Recurrent Neural Networks

ZHIBIN QUAN[1], ZHIQIANG GAO[1], WEILI ZENG[2], XUELIAN LI[1], AND MAN ZHU[3]
[1]School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
[2]College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[3]School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

Corresponding author: Zhibin Quan (sigma_quan@seu.edu.cn)

This work was supported in part by the National High Technology Research and Development Program (863 Program) of China under Grant 2015AA015406 and in part by the National Natural Science Foundation of China under Grant 61403081, Grant 61502095, Grant 61602260, Grant 61170165, and Grant 61702279.

ABSTRACT Memory-augmented recurrent neural networks (M-RNNs) have demonstrated empirically that they are very attractive for many applications, but a good theoretical understanding of their behaviors is unclear yet. In this paper, three analytical indicators named duration, addressability, and capacity of general forms of the additional memory in M-RNNs are formalized. The analysis results of the interactions among these indicators reveal that it is hard for an M-RNN to simultaneously provide good performance on more than two out of three of indicators. Meanwhile, the duration, addressability, and capacity are applied to analyze and compare two M-RNNs: long short term memory and neural turing machine for different cases. The comparison results show that none of the models has better performance on one indicator than the other model all the time. Moreover, it is found that separating memory system into sub-memories can bring the increasing duration and addressability and the decreasing capacity for the whole memory system.

INDEX TERMS Memory, recurrent neural networks, duration, addressability, capacity.

I. INTRODUCTION

Recurrent neural networks (RNNs) [1]–[3] are a rich family of neural networks with recurrent connections, as shown in the left part of figure 1. These recurrent connections make RNNs pose internal state that can memorize context information, enabling it to capture temporal correlations in sequential data. However, the memory which is implicitly encoded by hidden states and recurrent connections of RNNs, is limited. This limitation can be described from two perspectives. One is that the duration of its memory is often short because of the problems of vanishing and exploding gradients [4], [5]. The other is the limited capacity and the high complexity of memory accessing for the reason that the number of trainable parameters grows quadratically with the size of its memory [6].

Recent years, many research works focus on extending the memory of RNNs by introducing an additional memory which can be read and written. In this study, we refer to this class of models as memory-augmented recurrent neural networks (M-RNNs). As shown in figure 1, on the left is a RNN, and on the right is a M-RNN. Analogous to the recurrent connections in the hidden layer of RNN, the additional memory block of M-RNN is design to store temporal information, but with longer duration and larger capacity. Prime examples of M-RNN include long short term memory (LSTM) [7], RNNsearch [8], end-to-end memory networks (MemN2Ns) [9], neural Turing machines (NTMs) [10], dynamic memory networks (DMN) [11], fast associative memory [12] and neural semantic encoders (NSEs) [13]. These M-RNNs achieve success in many machine learning tasks, such as machine translation [8], [14], speech recognition [15], [16], image captioning [17], [18], question answering [11], [19], visual question answering [20], dialogue response generation [21], end-to-end dialogue [22] and action recognition [23].

In contrast to the empirical success of M-RNNs, the theoretical analysis and research works regarding M-RNNs are relatively lacking. To our knowledge, there are two properties which have been analyzed by the previous study [6], [24], [25]. On one hand, the size of the additional memory is often independent to the trainable parameters of RNNs [6]. This is equivalent to increasing the capacity of the memory in RNNs. On the other hand, once a piece of information is stored in the memory, it will be copied from time step to time

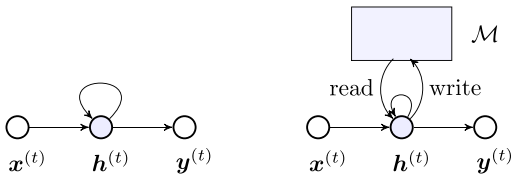


FIGURE 1. Schematic structures of a recurrent neural network (RNN) and a memory-augmented recurrent neural network (M-RNN). On the left is a RNN. On the right is a M-RNN. The recurrent connections in the hidden layer of RNN and the memory block in M-RNN allow information to persist from one input to another.

step and thus can stay there for a very long time [24]. This is an equivalent way of creating shortcut connections as pointed out by [25]. As a result, it can reduce the effect of vanishing and exploding gradients.

These various extensions and qualitative studies imply the following three indicators which can be used for understanding and measuring M-RNNs.

- *Duration.* How long can the content of memory persist?
- *Addressability.* How to measure the complexity of memory accessing?
- *Capacity.* How much (many) information in the memory can be used over a period of time?

Motivated by a desire to the question that "*can we design a M-RNN, which simultaneously provides good performance on all of the above three indicators for its memory?*", this paper makes an attempt to do quantitative analysis of the three indicators of M-RNNs, and to discuss the interactions among them.

To address this, we firstly derive a general definition of M-RNNs based on LSTM and NTM. Then, we give the definitions of the three indicators: duration, addressability and capacity (DAC) of M-RNNs. With careful derivation, we find that the three indicators can be characterized as functions of the size and updating frequency of the additional memory. This characteristic offers the bridge for analyzing the interactions among these indicators. Finally, by checking these interactions, a principle (called DAC principle) which can answer the above question, is discovered. This principle reveals that it is hard for a M-RNN to simultaneously provide good performance on more than two out of three of indicators.

In applications of DAC, we firstly derive minimum memory requirements for M-RNN when learning long term dependencies (LTDs) in sequential data; then we compare LSTM with NTM on the three indicators. The comparison results show that NTM does not have better performance on duration (or capacity, or addressability) than LSTM in all the cases, and vice versa. Additionally, we investigate the impacts when separating memory system into sub-memories.

The main contributions of this work are summarized as follows.

- We propose a general definition of M-RNNs which covers internal M-RNNs (e.g. LSTM) and external M-RNNs (e.g. NTM) as special cases.
- We propose three indicators (called *duration*, *capacity* and *addressability*), and use them to help understand and measure M-RNNs.
- We find out a principle (called DAC principle) which reveals that it is hard for a M-RNN to simultaneously provide good performance on more than two out of three of indicators.
- Further more, we prove that separating memory system into sub-memories can bring the increasing duration and addressability, and the decreasing capacity for the whole memory system.

The rest of this paper is organized as follows. In section II, we introduce the vanilla RNN and its two variants: LSTM and NTM. We then derive a generalized definition of M-RNN in section III which covers LSTM and NTM as special cases. In section IV, we propose definitions of the three indicators: duration, addressability, and capacity, and get the DAC principle by investigating and deriving the interactions among these indicators. We analyze the nature of LTDs and derive the minimum memory requirements for learning LTDs with M-RNN in section V. Additionally, we calculate the values of DAC for LSTM and NTM, and compare them on DAC under different settings. Moreover, we investigate how the values of DAC change when memory system is separated into a group of sub-memories in section VI. Finally, the last section gives conclusion and future work.

II. RECURRENT NEURAL NETWORKS

In this section we describe the vanilla RNN and its two variants, i.e. LSTM and NTM.

A vanilla RNN is a function $\mathcal{R} : X \times H \to Y \times H$, where X and Y are respectively input and output space; H is the hidden state space. On a variable-length input sequence $(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) \in X^T$ and with an initial hidden state $\boldsymbol{h}^{(0)} \in H$, the RNN transitions internally into states $(\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(T)})$ and outputs a sequence $(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(T)})$ according to

$$(\boldsymbol{y}^{(t)}, \boldsymbol{h}^{(t)}) = \mathcal{R}(\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}|\theta), \tag{1}$$

where θ denotes the set of trainable parameters (i.e. weight matrices and biases).

Long short term memory (LSTM) [7] is a particular variant of RNN's hidden unit. It has two types of hidden variables $\boldsymbol{c}^{(t)}$ and $\boldsymbol{h}^{(t)}$, where $\boldsymbol{c}^{(t)}$ is the "memory" cell which is designed to be maintained for a long time when necessary. The function of LSTM [26]

$$(\boldsymbol{y}^{(t)}, \boldsymbol{c}^{(t)}, \boldsymbol{h}^{(t)}) = \mathcal{R}(\boldsymbol{x}^{(t)}, \boldsymbol{c}^{(t-1)}, \boldsymbol{h}^{(t-1)}|\theta) \tag{2}$$

is defined as follows:

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh(W_{xc}\boldsymbol{x}^{(t)} + W_{hc}\boldsymbol{h}^{(t-1)} + b_c), \tag{3}$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \tilde{\boldsymbol{c}}^{(t)}, \tag{4}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)}), \tag{5}$$

where $\odot$ is element product; the output function is omitted; $\boldsymbol{i}, \boldsymbol{f}$, and $\boldsymbol{o}$ are the *input* gates, *forget* gates and *output* gates,

respectively. These gates are defined by

$$\boldsymbol{v}^{(t)} = \sigma(W_{xv}\boldsymbol{x}^{(t)} + W_{hv}\boldsymbol{h}^{(t-1)} + W_{cv}\boldsymbol{c}^{(t-1)} + b_v), \quad (6)$$

where $v \in \{i, f, o\}$; σ is the logistic function. All the gates are the same size as the hidden vector $\boldsymbol{h}$, and memory cell $\boldsymbol{c}$. The weight matrices W_{cv} are diagonal.

Neural Turing machine (NTM) [10], or its latest version Differentiable Neural Computer (DNC) [27] extends RNNs by introducing an external memory M. Like a digital computer, it has read / write heads for explicitly reading content from or writing content into M dynamically. The function of NTM

$$(\boldsymbol{y}^{(t)}, M^{(t)}, \boldsymbol{h}^{(t)}) = \mathcal{R}(\boldsymbol{x}^{(t)}, M^{(t-1)}, \boldsymbol{h}^{(t-1)}|\theta) \quad (7)$$

is defined by

$$M^{(t)} = M^{(t-1)} \odot (E - \boldsymbol{w}^{(t)}\boldsymbol{e}^{(t)}) + \boldsymbol{w}^{(t)}\boldsymbol{a}^{(t)}, \quad (8)$$

$$\boldsymbol{r}^{(t)} = M^{(t)}\boldsymbol{w}^{(t)}, \quad (9)$$

$$\boldsymbol{y}^{(t)} = \phi(\boldsymbol{h}^{(t)}, \boldsymbol{r}^{(t)}),$$

where the update equation of $\boldsymbol{h}$ is omitted; E is an all-ones matrix; $\boldsymbol{e}$ and $\boldsymbol{a}$ are the parameters for writing M; $\boldsymbol{r}$ is the read vector; $\boldsymbol{w}$ is the address for reading from (or writing into) M. $\boldsymbol{w}$ is calculated by an addressing function,

$$\boldsymbol{w}^{(t)} = \alpha(M^{(t-1)}, k), \quad (10)$$

where k is the query key.

III. MEMORY AUGMENTED RECURRENT NEURAL NETWORKS

Vanilla RNNs governed by equation (1) and its two extensions LSTM in equation (2) and NTM in equation (7) contain two types of memory: 1) internal memory i.e. the hidden state vector $\boldsymbol{h}^{(t-1)}$ or $\boldsymbol{c}^{(t-1)}$, where the memory size depends on the size of trainable parameters of RNN; 2) external memory $M^{(t)}$, where the memory size is independent to the size of trainable parameters of RNN. This section tries to derive a generalized definition for these extensions. For convenience, some notations are given in table 1.

Definition 1 (Memory-Augmented Recurrent Neural Networks (M-RNNs)): Let $M = \{m_i \in E | i = 1, 2, \ldots, N\}$ be a set of memory elements, where E can be any space of vector or matrix and N is the size of memory. If there are a read function $\gamma : E^N \times \mathbb{R}^N \mapsto E$, a write function $\omega : E^N \times \mathbb{R}^N \times E^C \mapsto E^N$, and an addressing function $\alpha : E^N \times E \mapsto \mathbb{R}^N$, such that given a context (or query) key $k \in E$ that is produced by a RNN $\mathcal{R}$, and a memory set M, we can get the address vector $\boldsymbol{w} \in \mathbb{R}^N$ by $\boldsymbol{w} \leftarrow \alpha(M, k)$ and read information $r \in E$ by $r \leftarrow \gamma(M, \boldsymbol{w})$, and the memory is updated by $M' \leftarrow \omega(M, \boldsymbol{w}, \boldsymbol{a})$, where $\boldsymbol{a} \in E^C$ is the content vector that is produced by $\mathcal{R}$, then the quadruple $\langle M, \alpha, \gamma, \omega \rangle$ is called an augmented memory $\mathcal{M}$ of $\mathcal{R}$, and $\mathcal{R}$ is called a memory-augmented RNN, which is re-denoted as $\mathcal{M}_{\mathcal{R}}$.

Figure 2 visualizes this definition, where $\boldsymbol{s}$ denotes all the state variables. Comparing with a vanilla RNN, $\mathcal{M}_{\mathcal{R}}$ has an additional quadruple $\langle M, \alpha, \gamma, \omega \rangle$. At each time step t, firstly, $\mathcal{M}_{\mathcal{R}}$ receives input $\boldsymbol{x}^{(t)}$, and combines it with $\boldsymbol{h}^{(t-1)}$ to produce an updated hidden state $\boldsymbol{h}^{(t)}$ and parameters $(k^{(t)}, \boldsymbol{a}^{(t)})$ which are used to access memory M; then the address $\boldsymbol{w}^{(t)}$ is calculated by the addressing function α and memory $M^{(t-1)}$ is updated by the write function ω; finally, $\mathcal{M}_{\mathcal{R}}$ combines $\boldsymbol{h}^{(t)}$ with an additional read vector $\boldsymbol{r}^{(t)}$ that is produced by the read function γ to produce the output $\boldsymbol{y}^{(t)}$.

TABLE 1. Notations for M-RNN and DAC.

Symbol	Description
$\boldsymbol{x}^{[1:\tau]}$	a variant length input sequence, where τ is the sequence length
$\boldsymbol{d}^{[1:\tau]}$	a variant length desired output sequence, where τ is the sequence length
$\mathcal{R}$	a recurrent neural network (RNN)
$\mathcal{M}_{\mathcal{R}}$	a memory-augmented RNN (M-RNN)
$\mathcal{M}$	an augmented memory, which is a quadruple $\langle M, \alpha, \gamma, \omega \rangle$
M	the memory set of $\mathcal{M}$
α	the address function of $\mathcal{M}$
γ	the read function of $\mathcal{M}$
ω	the write (update) function of $\mathcal{M}$
$\mathbb{D}(\mathcal{M})$	the duration of $\mathcal{M}$
$\mathbb{A}(\mathcal{M})$	the addressability of $\mathcal{M}$
$\mathbb{C}(\mathcal{M})$	the capacity of $\mathcal{M}$
$p(\omega)$	the frequency of wirte function ω
f_v	the funtion of duration, addressability and capacity, where $v \in \{D, A, C\}$
$T_y(t)$	the dependency span of the output $\boldsymbol{d}^{(t)}$
D	the dependency matrix
$T(t)$	the set of time step of some inputs, where the current desired output $\boldsymbol{d}^{(t)}$ is dependent on those inputs
$\kappa(t)$	the size of set $T(t)$
$N^*(t)$	the total memory demand at time step t
$\mathcal{M}'$	a separation of $\mathcal{M}$, e.g. $\mathcal{M}' = \{\mathcal{M}^{(1)}, \mathcal{M}^{(2)}\}$

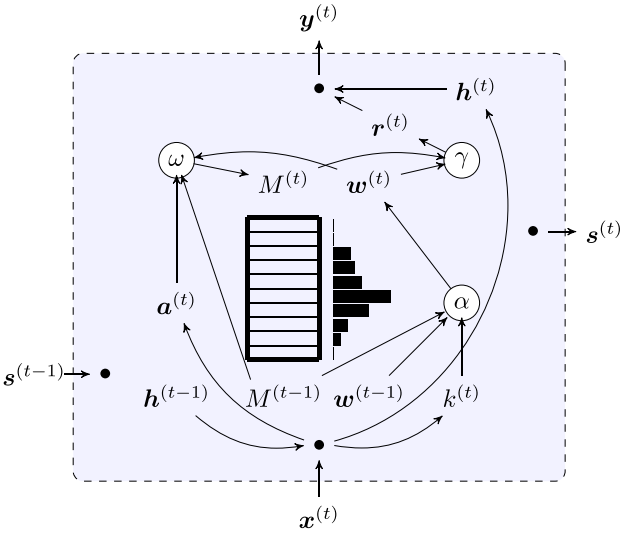


FIGURE 2. The generalized framework of memory-augmented recurrent neural networks (M-RNNs). Comparing with a vanilla RNN, a M-RNN $\mathcal{M}_{\mathcal{R}}$ has an additional quadruple $\langle M, \alpha, \gamma, \omega \rangle$, where M is a memory set; α is an addressing function; γ is a read function; and ω is a write function.

If the $\boldsymbol{w}$ satisfies the condition of softmax distribution:

$$\mathcal{S}_N = \{\boldsymbol{w} \in \mathbb{R}^N : \boldsymbol{w}_i \in [0, 1], \sum_{i=1}^{N} \boldsymbol{w}_i = 1\}, \tag{11}$$

then α is called a soft addressing. If $\boldsymbol{w}$ satisfies the condition:

$$\mathcal{H}_N = \{\boldsymbol{w} \in \mathbb{R}^N : \boldsymbol{w}_i \in \{0, 1\}, \sum_{i=1}^{N} \boldsymbol{w}_i = 1\}, \tag{12}$$

then α is called a hard addressing. The soft addressing offers $O(N)$ time complexity of memory accessing and the hard addressing offers $O(1)$ time complexity of memory accessing [6].

IV. DURATION, ADDRESSABILITY, CAPACITY, AND DAC PRINCIPLE

In this section, we propose the definitions of the three indicators: duration, addressability, capacity for M-RNNs. With detail derivation, we prove that the three indicators can be characterized as functions of the size and updating frequency of the memory. Based on this characteristic, the DAC principle is discovered at the end of this section.

A. DURATION

Definition 2 (Duration): Given a $\mathcal{M}_{\mathcal{R}}$, the duration $\mathbb{D}$ of $\mathcal{M}$ is defined as the average time intervals on each element m_i between two successive write operations $\omega_{t,i}$ and $\omega_{t',i}$, i.e.

$$\mathbb{D}(\mathcal{M}) \equiv \lim_{\tau\to\infty} \frac{1}{N_\delta(\tau)} \sum_{t=1}^{\tau} \sum_{i=1}^{N} \delta\left(\omega_{t,i}, \omega_{t',i}\right), \tag{13}$$

where $N_\delta(\tau)$ is the number of total time intervals; $t' = \min_{t<\hat{t}\leq\tau} \hat{t}$; δ is defined as

$$\delta\left(\omega_{t,i}, \omega_{t',i}\right) \equiv \begin{cases} t' - t, & \text{if } \omega_{t,i} = 1 \wedge \omega_{t',i} = 1, \\ \tau - t, & \text{if } \omega_{t,i} = 1 \wedge \forall_{t<t'\leq\tau} \omega_{t',i} = 0, \\ 0, & \text{otherwise,} \end{cases}$$

where $\omega_{t,i} = 1$, if there is a write operation on m_i at time step t; $\omega_{t,i} = 0$, otherwise.

Lemma 1: For any $\mathcal{M}_{\mathcal{R}}$, suppose the duration $\mathbb{D}$ of $\mathcal{M}$ satisfies *Definition 2*, then $\mathbb{D}(\mathcal{M})$ is a function f_D of memory size N and write frequency $p(\omega)$:

$$\mathbb{D}(\mathcal{M}) \equiv f_D(N, p(\omega)),$$

and f_D holds following properties:

(1) $f_D(N, p(\omega))$ is a monotonically increasing function of N if $p(\omega)$ takes a fixed value; $f_D(N, p(\omega))$ is a monotonically decreasing function of $p(\omega)$, if N is fixed. More precisely, f_D can be written as,

$$f_D(N, p(\omega)) = \frac{N}{p(\omega)}, \tag{14}$$

where $0 < p(\omega) < 1$.

(2) when $p(\omega) \approx 0$, i.e. for all the memory element $m_i, i = 1, \ldots N$ over all the time steps $t = 1, \ldots, \tau$, each memory element m_i just be written only once, the upper bound of f_D is,

$$\max f_D(N, p(\omega)) = \infty.$$

(3) when $p(\omega) = 1$, the lower bound of f_D is,

$$\min f_D(N, p(\omega)) = N.$$

Proof: When $\tau \to \infty$, we have

$$N_\delta(\tau) = p(\omega)\tau, \tag{15}$$

and the sum of time intervals on each element m_i between two successive write operations over a period of time τ can be written as,

$$\sum_{t=1}^{\tau} \delta\left(\omega_{t,i}, \omega_{t',i}\right) = \tau. \tag{16}$$

Substitute equation (15) and equation (16) into equation (13), we have,

$$\begin{aligned} \mathbb{D}(\mathcal{M}) &\equiv \lim_{\tau\to\infty} \frac{1}{N_\delta(\tau)} \sum_{t=1}^{\tau} \sum_{i=1}^{N} \delta\left(\omega_{t,i}, \omega_{t',i}\right) \\ &= \lim_{\tau\to\infty} \frac{1}{p(\omega)\tau} \sum_{i=1}^{N} \sum_{t=1}^{\tau} \delta\left(\omega_{t,i}, \omega_{t',i}\right) \\ &= \lim_{\tau\to\infty} \frac{N\tau}{p(\omega)\tau} \\ &= \frac{N}{p(\omega)}. \end{aligned}$$

Thus f_D holds equation (14).

Two properties can be deduced directly from the equation: 1) $f_D(N, p(\omega))$ is a monotonically increasing function of N if $p(\omega)$ takes a fixed value; 2) $f_D(N, p(\omega))$ is a monotonically decreasing function of $p(\omega)$, if N is fixed.

In practical applications, N often takes fixed value. In order to calculate the bounds of duration, let's consider three extreme cases of $p(\omega)$:

Case 0: $p(\omega) = 0$: this case does not make any sense because the memory can not be written anymore in this case.

Case 1: $p(\omega) \approx 0$: for all the memory element $m_i, i = 1, \ldots N$ over all the time steps $t = 1, \ldots, \tau$, each memory element m_i just be written only once.

Case 2: $p(\omega) = 1$: for every time step t, there is a write operation $\omega_{t,i}$ to update the value of memory element m_i.

For any τ, we can find a multiple of N which satisfies $nN = \tau$. Then the $\mathbb{D}(\mathcal{M})$ can be written as blow,

$$\mathbb{D}(\mathcal{M}) = \begin{cases} 0, & \text{if } p(\omega) = 0, \\ \dfrac{N-1}{2} + (n-1)N, & \text{if } p(\omega) \approx 0, \\ \dfrac{1}{n}\dfrac{N-1}{2} + \dfrac{n-1}{n}N, & \text{if } p(\omega) = 1. \end{cases}$$

With respect to *Case 1*, $\mathbb{D}(\mathcal{M})$ can be rewritten as

$$\mathbb{D}(\mathcal{M}) = \frac{N-1}{2} + (n-1)N$$

$$= nN - \frac{N+1}{2}$$
$$= \tau - \frac{N+1}{2}.$$

Then, we can get the upper bound of f_D,

$$\max f_D(N, p(\omega)) = \lim_{\tau\to\infty}\left(\tau - \frac{N+1}{2}\right) = \infty.$$

With respect to *Case 2*, $\mathbb{D}(\mathcal{M})$ can be rewritten as

$$\mathbb{D}(\mathcal{M}) = \frac{1}{n}\frac{N-1}{2} + \frac{n-1}{n}N$$
$$= N - \frac{N+1}{2n}$$
$$= N - \frac{N(N+1)}{2\tau}.$$

Thus, the lower bound of f_D is,

$$\min f_D(N, p(\omega)) = \lim_{\tau\to\infty}\left(N - \frac{N(N+1)}{2\tau}\right) = N.$$

■

B. ADDRESSABILITY

Definition 3 (Addressability): Given a $\mathcal{M}_{\mathcal{R}}$, the addressability $\mathbb{A}$ of $\mathcal{M}$ is defined as a monotonically decreasing function of the sum of the time complexity O_α, O_γ and O_ω of the algorithm of addressing function α, read function γ and write function ω on the memory set, i.e.

$$\mathbb{A}(\mathcal{M}) \equiv f(2O_\alpha + O_\gamma + O_\omega), \tag{17}$$

where f can be any monotonically decreasing function, such as $f(x) = -x^c$, where c ($c \geq 1$) is a constant.

The *Definition 3* reflects the average time complexity of memory accessing for the whole system.

Lemma 2: For any $\mathcal{M}_{\mathcal{R}}$, suppose the addressability of $\mathcal{M}$ satisfies *Definition 3*, then $\mathbb{A}(\mathcal{M})$ is a monotonically decreasing function f_A of memory size N.

Proof: From previous study [6], we have $O_\alpha = N$ and

$$O_\gamma = O_\omega = \begin{cases} N, & \text{if } \alpha \in \mathcal{S}_N, \\ 1, & \text{if } \alpha \in \mathcal{H}_N, \\ g(N), & \text{otherwise.} \end{cases}$$

The $g(N)$ is another monotonically increasing function of N, such as $\log(N)$ in [28]. $\mathbb{A}$ can be written as

$$\mathbb{A}(\mathcal{M}) = f_A(N) = \begin{cases} f(4N), & \text{if } \alpha \in \mathcal{S}_N, \\ f(2N+2), & \text{if } \alpha \in \mathcal{H}_N, \\ f(2N+2g(N)), & \text{otherwise.} \end{cases}$$

Because f is a monotonically decreasing function, f_A is a monotonically decreasing function of memory size N as well. ■

C. CAPACITY

Definition 4 (Capacity): Given a $\mathcal{M}_{\mathcal{R}}$, the capacity $\mathbb{C}$ of $\mathcal{M}$ is defined as the total bits of available information from $\mathcal{M}$ over a period of time Δ_τ, i.e. $[\tau, \tau+\Delta_\tau]$. Let each element m in memory set M contain k ($k \geq 1$) bits of information, i.e. $I(m) = k$, then the capacity $\mathbb{C}$ is defined as below,

$$\mathbb{C}(\mathcal{M}) \equiv \lim_{\tau\to\infty} I(m)(N + \lim_{\Delta_\tau\to\infty}\sum_{t=\tau}^{\tau+\Delta_\tau}\sum_{i=1}^{N}\omega_{t,i})$$
$$= \lim_{\tau\to\infty} k(N + \lim_{\Delta_\tau\to\infty}\sum_{t=\tau}^{\tau+\Delta_\tau}\sum_{i=1}^{N}\omega_{t,i}).$$

Lemma 3: For any $\mathcal{M}_{\mathcal{R}}$, suppose the capacity $\mathbb{C}$ of $\mathcal{M}$ satisfies *Definition 4*, then $\mathbb{C}(\mathcal{M})$ is a function f_C of memory size N and write frequency $p(\omega)$. More precisely, f_C can be written as,

$$\mathbb{C}(\mathcal{M}) \equiv f_C(N, p(\omega))$$
$$= kN + kp(\omega)\Delta_\tau. \tag{18}$$

Proof: When $\tau \to \infty$, we have $N_\delta(\tau) > N$. This means that at time step τ the total available information is kN. In the time steps from τ to $\tau + \Delta_\tau$, the amount of new available information depends on the number of write operations, and the number of write operations depends on the write frequency $p(\omega)$, we have

$$\lim_{\Delta_\tau\to\infty}\sum_{t=\tau}^{\tau+\Delta_\tau}\sum_{i=1}^{N}\omega_{t,i} = \lim_{\Delta_\tau\to\infty} p(\omega)\Delta_\tau.$$

Thus f_C holds equation (18). ■

In a standard information storage system, e.g., the disk of a digital computer, the amount of information storage is an explicit quantity. However, *Lemma 3* reveals that the capacity which is defined by the *Definition 4* reflects a key insight: the updating frequency of memory can increase the capacity for the whole system.

D. DAC PRINCIPLE

Based on *Definition 2*, *3* and *4*, we can find the following principle for M-RNNs.

Theorem 1 (Duration, Addressability and Capacity (DAC) Principle): It is hard for a $\mathcal{M}_{\mathcal{R}}$ simultaneously provide good performance on more than two out of three of the indicators: $\mathbb{D}$, $\mathbb{A}$ and $\mathbb{C}$.

Proof: From the *Lemma 1*, *2* and *3*, we have:

(1) Duration $\mathbb{D}$ is a function f_D of memory size N and write frequency $p(\omega)$. If N is fixed, then f_D is a monotonically decreasing function of $p(\omega)$; If $p(\omega)$ is fixed, then f_D is a monotonically increasing function of N; If N and $p(\omega)$ increase (or decrease) simultaneously, then the change of f_D is uncertain. Figure 3 (a) illustrates these properties.

(2) Addressability $\mathbb{A}$ is a monotonically decreasing function f_A of memory size N as shown in figure 3 (b).

(3) Capacity $\mathbb{C}$ is a function f_C of memory size N and write frequency $p(\omega)$. If N is fixed, then f_C is a monotonically increasing function of $p(\omega)$; If $p(\omega)$ is fixed,

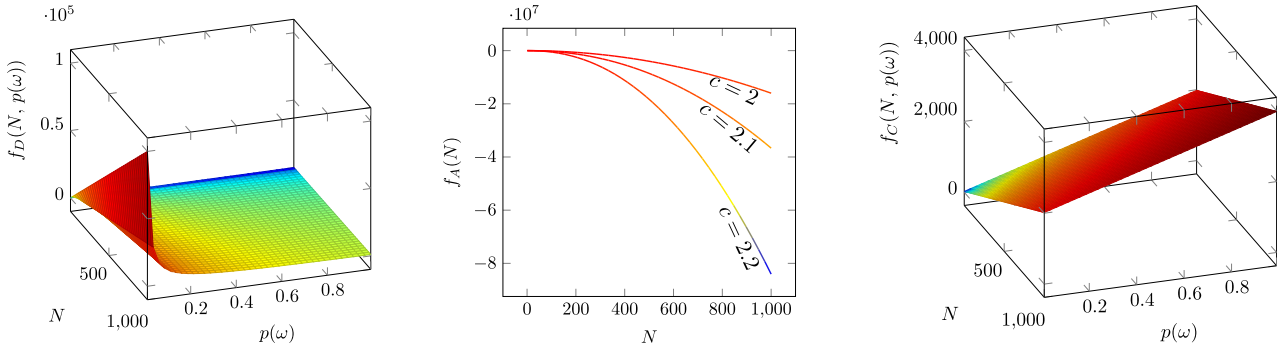


FIGURE 3. Visualizing the functions (duration f_D, addressability f_A, and capacity f_C). In the all sub-figures, axis N represents the memory size; axis $p(\omega)$ represents the write frequency. (a) Duration $f_D(N, p(\omega)) = N/p(\omega)$. (b) Addressability $f_A(N) = -(4N)^c$. (c) Capacity $f_C(p(\omega)) = kN + kp(\omega)\Delta_\tau$, where $k = 2$ and $\Delta_\tau = 1000$.

TABLE 2. Analysis on the variational laws of duration $\mathbb{D}$, addressability $\mathbb{A}$ and capacity $\mathbb{C}$ with respected to memory size N and write frequency $p(\omega)$. '–' means fixed or no change; '↑' means increasing; '↓' means decreasing; '⇕' means increasing or decreasing or no change; "Var." shorts for variable.

Case	Independent Var.				Dependent Var.		
					$\mathbb{D}$	$\mathbb{A}$	$\mathbb{C}$
1	$p(\omega)$	–	N	↑	↑	↓	↑
2				↓	↓	↑	↓
3	N	–	$p(\omega)$	↑	↓	–	↑
4				↓	↑	–	↓
5	$p(\omega)$	↑	N	↑	⇕	↓	↑
6				↓	↓	↑	⇕
7	N	↑	$p(\omega)$	↓	↑	↓	⇕
8	$p(\omega)$	↓	N	↓	⇕	↑	↓
9	N	–	$p(\omega)$	–	–	–	–

then f_C is a monotonically increasing function of N; If N and $p(\omega)$ increase (or decrease) simultaneously, then f_C is increasing (or decreasing). Figure 3 (c) illustrates these properties.

There are nine cases in total, because both N and $p(\omega)$ have three possibilities: fixed: '−', increasing: '↑' and decreasing: '↓'. It is easy to identify examples of each pairing of DAC, outlining the proof by exhaustive example of DAC principle in table 2. For example, in case 1 in the table, $p(\omega)$ is fixed: '−'; when N is increasing: '↑', the value of $\mathbb{D}$ is increasing: '↑'; $\mathbb{A}$ is decreasing: '↓'; and $\mathbb{C}$ is increasing: '↑'. None of cases in table 2 has increasing '↑' values of all the three indicators. ■

Theorem 1 implies that *the three indicators can be characterized as functions of the size N and updating (write) frequency $p(\omega)$ of the additional memory*. This offers two insights for helping design and train a M-RNN:

- Tuning the hyper-parameter (memory size N) can change the values of DAC.
- Different training algorithm can force M-RNNs to learn different write frequency $p(\omega)$. This leads different values of DAC.

V. APPLICATIONS OF DAC

This section uses the DAC to help design M-RNNs for learning long term dependencies in sequential data, and to compare two M-RNNs: LSTM and NTM.

A. DAC FOR HELPING DESIGN M-RNNs FOR LEARNING LONG TERM DEPENDENCIES

Long term dependency (LTD) [4], also called long memory process [29], [30] or long range dependency (persistence) [31], [32], is a phenomenon that may arise in the analysis of spatial or time series data. It relates to the dependency of two points with long time interval or spatial distance between the points.

Definition 5 (Long Term Dependencies (LTDs) [4]): If the prediction of the desired output $\boldsymbol{d}^{(t_i)}$ of a sequence-to-sequence task $\mathcal{T}$ at time step t_i depends on the inputs $\{\boldsymbol{x}^{(t_{j_1})}, \boldsymbol{x}^{(t_{j_2})}, \ldots, \boldsymbol{x}^{(t_{j_{\kappa(t_i)}})}\}$ presented at earlier time steps $T(t_i) = \{t_{j_1}, t_{j_2}, \ldots, t_{j_{\kappa(t_i)}}\}$, and

$$\min T(t_i) \ll t_i,$$

where $\kappa(t_i)$ is the size of dependency time steps at time step t_i and $\min T(t_i)$ is the time step of the input which is the longest dependency of the output $\boldsymbol{d}^{(t_i)}$, then the task $\mathcal{T}$ displays long term dependencies.

If a sequencial processing task $\mathcal{T}$ displays LTDs, learning $\mathcal{T}$ using vanilla RNNs with stochastic gradient descent is a challenge due to the vanish gradient problem [4], [5]. M-RNNs can avoid this problem by writing the relevant inputs into and reading them from the additional memory when necessary. Here, we aim to use the DAC to analyze the nature of LTDs, as well as the requirements when using M-RNNs to learn LTDs in sequencial data.

Definition 6 (Dependency Span): If an output $\boldsymbol{d}^{(t)}$ of a sequence-to-sequence task $\mathcal{T}$ at time step t depends on the inputs presented at earlier time steps $T(t) = \{t_{j_1}, t_{j_2}, \ldots, t_{j_{\kappa(t)}}\}$, then the dependency span of the output $\boldsymbol{d}^{(t)}$ is defined as

$$T_y(t) = t - \min T(t),$$

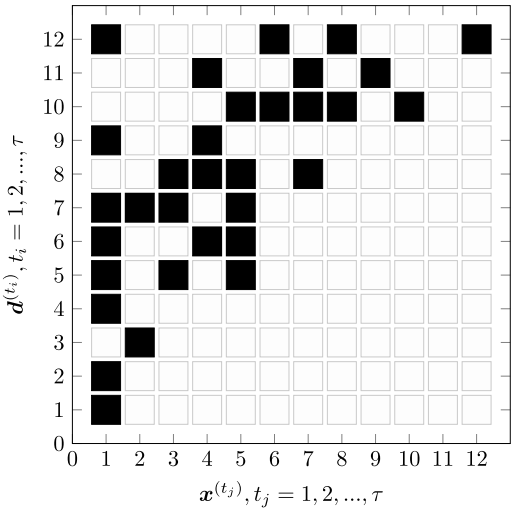


FIGURE 4. Illustration of the dependency matrix D of a sequential pair $(x^{[1:\tau]}, d^{[1:\tau]})$, where $\tau = 12$. Each square represents a $D_{i,j}$ (black for one and white for zero) in *Definition 7*. The horizontal axis represents the input sequence and the vertical axis represents the output sequence.

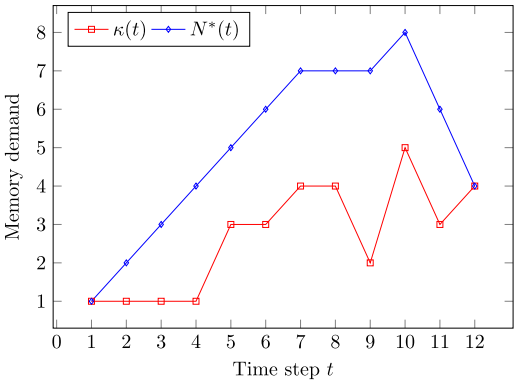


FIGURE 5. Curves of memory demand when learning sequential data. $\kappa(t)$ is the size of dependent inputs of current output $d^{(t)}$, which equals to the minimum memory demand for making the prediction at time step t; $N^*(t)$ is the total memory demand at time step t.

where $\min T(t)$ is the time step of the input which is the longest dependency of the current output $\boldsymbol{d}^{(t)}$.

Definition 7 (Dependency Matrix): A dependency matrix is used to denote the dependencies over a period of time steps $\{1, 2, \ldots, \tau\}$ of two sequences $\boldsymbol{d}^{[1:\tau]}$, $\boldsymbol{x}^{[1:\tau]}$, i.e. $D = \{D_{i,j} | 1 \le i \le \tau, 1 \le j \le \tau\}$. Each element of the matrix is defined as flow,

$$D_{i,j} = \begin{cases} 1, & \text{if } \boldsymbol{d}^{(i)} \text{ dependents on } \boldsymbol{x}^{(j)}, \\ 0, & \text{otherwise.} \end{cases}$$

Here, we propose the definition of dependency matrix just for analyzing the nature of LTDs and for helping design a $\mathcal{M}_{\mathcal{R}}$ to capture the LTDs. In practical applications, e.g. machine translation, the exact values of a particular dependency matrix are actually unknown, but the expectation value of the dependency matrix can be estimated from data.

A dependency matrix is usually a upper triangular matrix because the current output $\boldsymbol{d}^{(t_i)}$ always depends on the inputs $\boldsymbol{x}^{(t_j)}$ at previous time steps ($t_j \le t_i$). Figure 4 visualizes the 12×12 dependency matrix D of a sequential pair $(\boldsymbol{x}^{[1:12]}, \boldsymbol{d}^{[1:12]})$.

Consider a sequence-to-sequence task $\mathcal{T}$ which displays LTDs. The input sequence is $\boldsymbol{x}^{[1:\tau]}$ and the corresponding desired output sequence is $\boldsymbol{d}^{[1:\tau]}$. Let D be their dependency matrix. For any time step t, $1 \le t \le \tau$, the $\kappa(t)$ can be calculated by the following equation,

$$\kappa(t) = \sum_{t_j=1}^{\tau} D_{t,t_j},$$

and the dependency time steps set $T(t)$ can be calculated as,

$$T(t) = \left\{ t_j | D_{t,t_j} = 1, 1 \le t_j \le \tau \right\}.$$

For example, consider a sequence pair $(\boldsymbol{x}^{[1:12]}, \boldsymbol{d}^{[1:12]})$ which is governed by the dependency matrix as shown in figure 4, we can get $\kappa(1) = 1$, $\kappa(12) = 4$, $T(1) = \{1\}$, $T(12) = \{1, 6, 8, 12\}$, $T_y(1) = 1 - 1 = 0$ and $T_y(12) = 12 - 1 = 11$.

We can also calculate the duration $\mathbb{D}_x$ demand of each input $\boldsymbol{x}^{(t)}$, $1 \le t \le \tau$ with D,

$$\mathbb{D}_x(t) = \mathbb{D}_x^+(t) - t.$$

where $\mathbb{D}_x^+(t)$ is defined as,

$$\mathbb{D}_x^+(t) = \max \left\{ t_i | D_{t_i,t} = 1, 1 \le t_i \le \tau \right\}.$$

In figure 4, we can get $\mathbb{D}_x(1) = 12 - 1 = 11$, and $\mathbb{D}_x(12) = 12 - 12 = 0$. With the duration demand of each input, the total memory demand over a period of time can be calculated as,

$$N^*(t) = \sum_{t_j=1}^{t} \mathbb{I}(\mathbb{D}_x^+(t_j) \ge t),$$

where $\mathbb{I}$ is the indicator function. In figure 4, $N^*(1) = 1$ and $N^*(12) = 1+0+0+0+0+1+0+1+0+0+0+1 = 4$.

With the calculations of $N^*(t)$ and $\kappa(t)$, we have

$$N^*(t) \ge \kappa(t), \text{ for } t = 1, 2, \ldots, \tau.$$

Figure 5 shows the values of $N^*(t)$ and $\kappa(t)$ for the sequence pair $(\boldsymbol{x}^{[1:12]}, \boldsymbol{d}^{[1:12]})$ which is governed by the dependency matrix as shown in figure 4. $N^*(t)$ and $\kappa(t)$ offer two conditions for designing a $\mathcal{M}_{\mathcal{R}}$.

Condition 1 (Minimum Memory Requirement for Whole Sequence Prediction): Suppose a $\mathcal{M}_{\mathcal{R}}$ is used to capture the LTDs in a sequence pair $(\boldsymbol{x}^{[1:\tau]}, \boldsymbol{d}^{[1:\tau]})$ which is govern by a dependency matrix D, then the memory size N of $\mathcal{M}_{\mathcal{R}}$ must hold that,

$$N \ge \max_{1 \le t \le \tau} N^*(t).$$

Condition 2 (Minimum Memory Requirement for Single Prediction): Suppose a $\mathcal{M}_{\mathcal{R}}$ is used to make prediction at time step t, then the memory size N of $\mathcal{M}_{\mathcal{R}}$ must hold that,

$$N \ge \max_{1 \le t \le \tau} \kappa(t).$$

In practical applications, the value of $N^*(t)$ is unknown, because the exactly durations $\mathbb{D}_x(t)$ of each input $\boldsymbol{x}^{(t)}$ are unknown. The value of $\kappa(t)$ is also unknown because the dependency time step set $T(t)$ of each output $\boldsymbol{d}^{(t)}$ is unknown.

B. DAC FOR COMPARING LSTM AND NTM

We need to calculate the DAC values for LSTM and NTM firstly, before using DAC to compare them.

1) CALCULATING THE DAC VALUES FOR LSTM

In order to calculate the DAC values for LSTM, let us analyze equivalences between LSTM and M-RNN.

Theorem 2 (A LSTM as a M-RNN): There exists equivalences in a LSTM, which correspond to the four tuples $\langle M, \alpha, \gamma, \omega\rangle$ in a $\mathcal{M}_{\mathcal{R}}$.

Proof: In order to prove the equivalence, we need to prove that there exists a ''memory'' M for storing information, an ''addressing'' function to obtain address, a ''write'' function to update the memory, and a ''read'' function to get information in a LSTM.

(1) Firstly, let memory cells $\boldsymbol{c}$ in LSTM be the M in $\mathcal{M}_{\mathcal{R}}$, i.e. $\boldsymbol{c} = M$.

(2) Secondly, let the gated functions of LSTM in equation (6) be a kind of addressing function α in $\mathcal{M}_{\mathcal{R}}$,

$$\begin{aligned}\boldsymbol{v}^{(t)} &= \sigma(k_v^{(t)} + W_{cv}M^{(t-1)} + b_v)\\ &= \alpha(M^{(t-1)}, k_v^{(t)}|\theta_v),\end{aligned}$$

where $k_v^{(t)} = W_{xv}\boldsymbol{x}^{(t)} + W_{hv}\boldsymbol{h}^{(t-1)} + b_v$ and $\theta_v = W_{cv}$.

(3) Let $\boldsymbol{i}^{(t)} \odot \tilde{\boldsymbol{c}}^{(t)}$ be the content $\boldsymbol{a}^{(t)}$ for updating memory, i.e. $\boldsymbol{a}^{(t)} = \boldsymbol{i}^{(t)} \odot \tilde{\boldsymbol{c}}^{(t)}$, then, the equation (4) can be regarded as the memory M write function ω,

$$\begin{aligned}M^{(t)} &= \boldsymbol{w}^{(t)} \odot M^{(t-1)} + \boldsymbol{a}^{(t)}\\ &= \omega(M^{(t-1)}, \boldsymbol{w}^{(t)}, \boldsymbol{a}^{(t)}),\end{aligned}$$

where $\boldsymbol{w}^{(t)} = \boldsymbol{f}^{(t)}$ is a write address.

(4) Finally, let $\boldsymbol{h}^{(t)}$ be the read content $\boldsymbol{r}^{(t)}$, then the equation (5) can be considered as a read function γ,

$$\boldsymbol{r}^{(t)} = \boldsymbol{w}^{(t)} \odot \tanh(M^{(t)}) = \gamma(M^{(t)}, \boldsymbol{w}^{(t)}),$$

where $\boldsymbol{w}^{(t)} = \boldsymbol{o}^{(t)}$ is a read address. ■

Let N_l be the size of memory of a LSTM $\mathcal{M}_{lstm}$, i.e. the hidden layer size is N_l and the write frequency is $p(\omega_l)$. We have following properties of $\mathcal{M}_{lstm}$.

Property 1: $\mathbb{D}(\mathcal{M}_{lstm}) = N_l/p(\omega_l)$.

Property 2: $\mathbb{A}(\mathcal{M}_{lstm}) = -(2N_l^2 + 2N_l)^c$.

Proof: From the proof of *Theorem 2* we can observe that: 1) the time complexity of addressing function in equation (6) is $O(N_l^2)$, because W_{hv} is $N_l \times N_l$ matrix; 2) the time complexities of read and write functions are both $O(N_l)$, because it requires accessing the whole M (i.e. $\boldsymbol{c}$) for the two functions. Thus, we have $O_\alpha = N_l^2$, $O_\gamma = N_l$ and $O_\omega = N_l$. With the *Definition 3*, *Property 2* is obtained, if we substitute these values into equation (17) and let $f(x) = -x^c$. ■

Property 3: $\mathbb{C}(\mathcal{M}_{lstm}) = N_l + p(\omega_l)\Delta_\tau$.

TABLE 3. Comparing LSTM with NTM on duration $\mathbb{D}$, addressability $\mathbb{A}$ and capacity $\mathbb{C}$ under the different settings of memory size N and write frequency $p(\omega)$. 'l' represents LSTM; 'n' represents NTM; '=' means equals to; '>' means greater than; '<' means less than; '<>' means uncertain.

Case	Condition		Indicator		
	N	$p(\omega)$	$\mathbb{D}$	$\mathbb{A}$	$\mathbb{C}$
1	$l = n$	$l = n$	$l = n$	$l < n$	$l = n$
2		$l > n$	$l < n$	$l < n$	$l > n$
3		$l < n$	$l > n$	$l < n$	$l < n$
4	$l > n$	$l = n$	$l > n$	$l < n$	$l > n$
5		$l > n$	$l <> n$	$l < n$	$l > n$
6		$l < n$	$l > n$	$l < n$	$l <> n$
7	$l < n$	$l = n$	$l < n$	$l <> n$	$l < n$
8		$l > n$	$l < n$	$l <> n$	$l <> n$
9		$l < n$	$l <> n$	$l <> n$	$l < n$

2) CALCULATING THE DAC VALUES FOR NTM

In order to calculate the DAC values for NTM, we need to analyze the equivalences between NTM and M-RNN.

Theorem 3 (A NTM as a M-RNN): There exists equivalences in a NTM, which correspond to the four tuples $\langle M, \alpha, \gamma, \omega\rangle$ in a $\mathcal{M}_{\mathcal{R}}$.

Proof: Let M in equation (7) be the memory set in *Definition 1*. Then, the equation (10), equation (9) and equation (8) can be considered as the addressing function α, the read function γ, and the write function ω respectively. ■

Let N_n be the size of memory of a NTM $\mathcal{M}_{ntm}$ and $p(\omega_n)$ be the write frequency. We have following properties of $\mathcal{M}_{ntm}$.

Property 4: $\mathbb{D}(\mathcal{M}_{ntm}) = N_n/p(\omega_n)$.

Property 5: $\mathbb{A}(\mathcal{M}_{ntm}) = -(4N_n)^c$.

Proof: We have $O_\alpha = N_n$, $O_\gamma = N_n$ and $O_\omega = N_n$ because the addressing function α of $\mathcal{M}_{ntm}$ is soft addressing mechanism. *Property 5* is obtained, if we substitute these values into equation (17) and let $f(x) = -x^c$. ■

Property 6: $\mathbb{C}(\mathcal{M}_{ntm}) = kN_n + kp(\omega_n)\Delta_\tau$.

3) COMPARISON BETWEEN LSTM AND NTM USING DAC

In order to compare LSTM with NTM using DAC, let each memory element in NTM contain one bit information, i.e. $k = 1$ and time intervals Δ_τ for capacity of two models are equal.

Table 3 shows the comparison results of DAC values of the two models under the different settings of memory size N and write frequency $p(\omega)$. For example, in case 1 in the table, if memory sizes are equal, i.e., $N_l = N_n$ and write frequencies are equal, i.e., $p(\omega_l) = p(\omega_n)$, then the durations are equal, i.e., $\mathbb{D}(\mathcal{M}_{lstm}) = \mathbb{D}(\mathcal{M}_{ntm})$; the addressability of LSTM is less than the addressability of NTM, i.e., $\mathbb{A}(\mathcal{M}_{lstm}) < \mathbb{A}(\mathcal{M}_{ntm})$; and the capacities are equal, i.e., $\mathbb{C}(\mathcal{M}_{lstm}) = \mathbb{C}(\mathcal{M}_{ntm})$.

Two observations can be deduced directly from the nine cases in table 3:

(1) $\mathcal{M}_{lstm}$ does not have better performance in $\mathbb{D}$ (or $\mathbb{A}$, or $\mathbb{C}$) than $\mathcal{M}_{ntm}$ in all the cases, and vice versa;

(2) $\mathbb{A}(\mathcal{M}_{ntm})$ is better than $\mathbb{A}(\mathcal{M}_{lstm})$ when $N_l \geq N_n$.

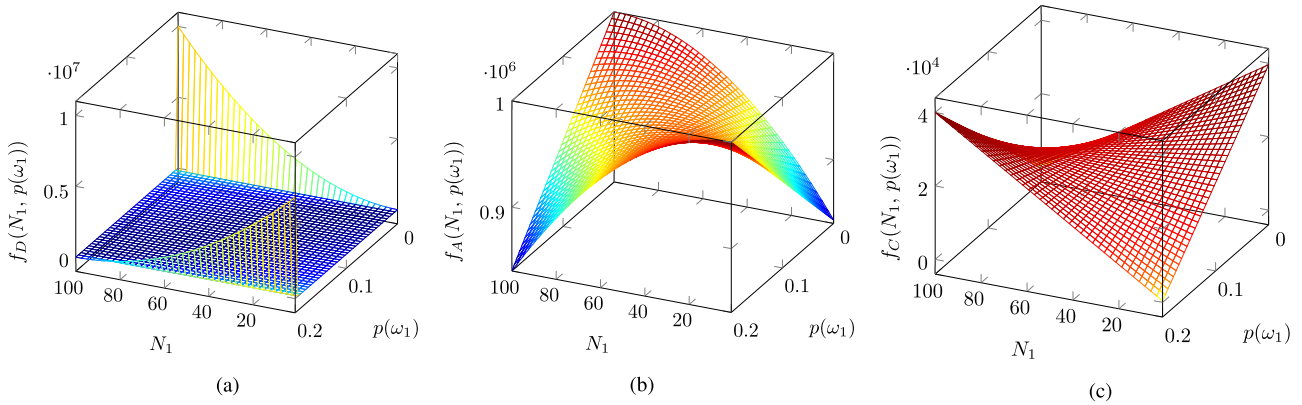


FIGURE 6. Visualizing the functions (duration f_D, addressability f_A and capacity f_C) of memory system $\mathcal{M}'$, where $\mathcal{M}' = \{\mathcal{M}^{(1)}, \mathcal{M}^{(2)}\}$ is a separation of $\mathcal{M}$. The duration f_D, addressability f_A and capacity f_C of $\mathcal{M}'$ are the functions of the size N_1 and the writing frequency $p(\omega_1)$ of its sub-memory $\mathcal{M}^{(1)}$. The values of DAC can be tuned by changing the value of N_1 or $p(\omega_1)$. (a) Duration. (b) Addressability, where $c = 2$. (c) Capacity, where $k = 2$ and $\tau = 10000$.

VI. DISCUSSION

We here try to discuss the problem: "how the values of DAC change when we separate the memory of a $\mathcal{M}$ into a group of sub-memories $\mathcal{M}^{(1)}, \mathcal{M}^{(2)}, \ldots, \mathcal{M}^{(n)}$?".

Suppose $\mathcal{M}'$ is a separation of $\mathcal{M}$, and it has two sub-memories $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$. Let N_1 and N_2 be their memory sizes, and $p(\omega_1)$ and $p(\omega_2)$ be their frequencies of write operation. For comparing $\mathcal{M}'$ with $\mathcal{M}$, we have following constraints:

$$N_1 \leq N_2, \tag{19}$$
$$p(\omega_2) \leq p(\omega_1), \tag{20}$$
$$N_1 + N_2 = N, \tag{21}$$
$$p(\omega_1) + p(\omega_2) = p(\omega), \tag{22}$$
$$\alpha_1, \alpha_2 \in \mathcal{S}_N. \tag{23}$$

$\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$ have following properties by *Lemmas 1 - 3*.

Property 7: $\mathbb{D}(\mathcal{M}^{(1)}) = N_1/p(\omega_1)$.
Property 8: $\mathbb{A}(\mathcal{M}^{(1)}) = -(4N_1)^c$.
Property 9: $\mathbb{C}(\mathcal{M}^{(1)}) = kN_1 + kp(\omega_1)\Delta_\tau$.
Property 10: $\mathbb{D}(\mathcal{M}^{(2)}) = N_2/p(\omega_2)$.
Property 11: $\mathbb{A}(\mathcal{M}^{(2)}) = -(4N_2)^c$.
Property 12: $\mathbb{C}(\mathcal{M}^{(2)}) = kN_2 + kp(\omega_2)\Delta_\tau$.

The properties of $\mathcal{M}'$ can be derived based on the properties of it sub-memories $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$.

Theorem 4: Let $\mathcal{M}'$ be a separation of $\mathcal{M}$, suppose its sub-memories satisfy the constraints (19) - (23), then this separation

- brings the increase of duration and addressability, and the decrease of capacity;
- and makes a trade-off among DAC when the sub-memory size N_1 (or N_2) is changed.

Proof: The DAC values of $\mathcal{M}'$ are the weighted sums of the DAC values of its two sub-memories $\mathcal{M}^{(1)}$ and $\mathcal{M}^{(2)}$.

$$\mathbb{D}(\mathcal{M}') = \frac{N_2}{N}\mathbb{D}(\mathcal{M}^{(2)}) + \frac{N_1}{N}\mathbb{D}(\mathcal{M}^{(1)});$$
$$\mathbb{A}(\mathcal{M}') = \frac{p(\omega_2)}{p(\omega)}\mathbb{A}(\mathcal{M}^{(2)}) + \frac{p(\omega_1)}{p(\omega)}\mathbb{A}(\mathcal{M}^{(1)});$$
$$\mathbb{C}(\mathcal{M}') = \frac{N_2}{N}\mathbb{C}(\mathcal{M}^{(2)}) + \frac{N_1}{N}\mathbb{C}(\mathcal{M}^{(1)}).$$

Substitute *Properties 7 - 12* and equality constraints (21) and (22) into the above equations, then,

$$\begin{aligned}
\mathbb{D}(\mathcal{M}') &\equiv f_D(N_1, p(\omega_1)) \\
&= \frac{(N - N_1)^2}{N(p(\omega) - p(\omega_1))} + \frac{N_1^2}{Np(\omega_1)}; \\
\mathbb{A}(\mathcal{M}') &\equiv f_A(N_1, p(\omega_1)) \\
&= -\frac{(p(\omega) - p(\omega_1))(4(N - N_1))^c}{p(\omega)} \\
&\quad - \frac{p(\omega_1)(4N_1)^c}{p(\omega)}; \\
\mathbb{C}(\mathcal{M}') &\equiv f_C(N_1, p(\omega_1)) \\
&= k\frac{N - N_1}{N}((N - N_1) + (p(\omega) - p(\omega_1))\Delta_\tau) \\
&\quad + k\frac{N_1}{N}(N_1 + p(\omega_1)\Delta_\tau).
\end{aligned}$$

The DAC values of $\mathcal{M}'$ are the functions of memory size N_1 and writing frequency $p(\omega_1)$ of $\mathcal{M}^{(1)}$, i.e., $f_v(N_1, p(\omega_1))$, $v \in \{D, A, C\}$. Figure 6 visualizes these functions. For all the sub-figures, the N in equality constraints (21) is 100 and the $p(\omega)$ in equality constraints (22) is 0.2. Consider the region which is constituted by inequality constraints (19) and (20), by checking the gradients of $f_v(N_1, p(\omega_1))$ with respect to N_1 and $p(\omega_1)$, we come to the conclusions:

(1) Comparing with $\mathcal{M}$,

$$\mathbb{D}(\mathcal{M}') > \mathbb{D}(\mathcal{M}),$$
$$\mathbb{A}(\mathcal{M}') > \mathbb{A}(\mathcal{M}),$$
$$\mathbb{C}(\mathcal{M}') < \mathbb{C}(\mathcal{M});$$

(2) Changing the value of N_1 (or N_2) can make a trade-off among the DAC values of $\mathcal{M}'$. ■

VII. CONCLUSION AND FUTURE WORK

In this study, we introduced a general framework for memory-augmented recurrent neural networks (M-RNNs) and proposed three indicators: duration, addressability, and capacity

for measuring and understanding the "memory" of M-RNN. To our knowledge this is the first work to understand M-RNNs from the perspective of these indicators. The three indicators are used to measure three aspects of M-RNNs: the duration of memory content, the complexity of memory accessing and the available information of memory over a period of time. Based on these indicators, the *DAC principle* is discovered, which reveals that it is hard for a M-RNN to simultaneously provide good performance on more than two out of three of indicators. The DAC is applied to help design M-RNNs for learning long term dependencies in sequential data, and to analyze and compare LSTM with NTM in different cases. The comparison results show that NTM does not have better performance on duration (or addressability, or capacity) than LSTM in all the cases, and vice versa. Moreover, we showed that separating a memory system into sub-memories can bring the increasing duration and addressability, and the decreasing capacity for the whole memory system.

As future work, the definition and calculation of DAC need more detail consideration; then, we should do quantitative and qualitative analysis for different read and write functions, and addressing functions of the memory; thirdly, study the impacts of different memory structure; finally, set different models with the guideline of DAC and test these models in a wide range of machine learning tasks. We also plan to verify the sub-memory property in a series of natural language processing tasks, such as relation extraction, semantic role labeling, language modeling, text comprehension, and question answering.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[2] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Netw.*, vol. 1, no. 4, pp. 339–356, Oct. 1988.

[3] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[4] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[5] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Mach. Learn.*, vol. 28. 2013, pp. 1310–1318.

[6] J. Rae *et al.*, "Scaling memory-augmented neural networks with sparse reads and writes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3621–3629.

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

[8] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, pp. 1–15, 2014. [Online]. Available: http://arxiv.org/abs/1409.0473

[9] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2440–2448.

[10] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *CoRR*, vol. abs/1410.5401, pp. 1–26, 2014. [Online]. Available: http://arxiv.org/abs/1410.5401

[11] A. Kumar *et al.*, "Ask me anything: Dynamic memory networks for natural language processing," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48. New York, NY, USA, Jun. 2016, pp. 1378–1387.

[12] J. Ba, G. E. Hinton, V. Mnih, J. Z. Leibo, and C. Ionescu, "Using fast weights to attend to the recent past," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4331–4339.

[13] T. Munkhdalai and H. Yu, "Neural semantic encoders," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, vol. 1. Valencia, Spain, Apr. 2017, pp. 397–407. [Online]. Available: http://www.aclweb.org/anthology/E17-1038

[14] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation.," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[15] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks.," in *Proc. ICASSP*, May 2013, pp. 6645–6649.

[16] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. 31st Int. Conf. Mach. Learn.*, vol. 32. Beijing, China, Jun. 2014, pp. 1764–1772.

[17] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37. Lille, France, Jul. 2015, pp. 2048–2057.

[18] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 652–663, Apr. 2017.

[19] J. Weston, A. Bordes, S. Chopra, and T. Mikolov, "Towards AI-complete question answering: A set of prerequisite toy tasks," *CoRR*, vol. abs/1502.05698, pp. 1–14, 2015. [Online]. Available: http://arxiv.org/abs/1502.05698

[20] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48. New York, NY, USA, Jun. 2016, pp. 2397–2406.

[21] I. V. Serban *et al.*, "Multiresolution recurrent neural networks: An application to dialogue response generation," in *Proc. AAAI*, 2017, pp. 3288–3294.

[22] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proc. AAAI*, 2016, pp. 3776–3784.

[23] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access*, vol. 6, pp. 1155–1166, 2017.

[24] S. Chandar, S. Ahn, H. Larochelle, P. Vincent, G. Tesauro, and Y. Bengio, "Hierarchical memory networks," *CoRR*, vol. abs/1605.07427, pp. 1–10, 2016. [Online]. Available: http://arxiv.org/abs/1605.07427

[25] Ç. Gülçehre, S. Chandar, and Y. Bengio, "Memory augmented neural networks with wormhole connections," *CoRR*, vol. abs/1701.08718, pp. 1–27, 2017. [Online]. Available: http://arxiv.org/abs/1701.08718

[26] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *J. Mach. Learn. Res.*, vol. 3, no. 1, pp. 115–143, 2003.

[27] A. Graves *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

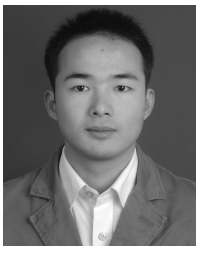
[28] M. Andrychowicz and K. Kurach, "Learning efficient algorithms with hierarchical attentive memory," *CoRR*, vol. abs/1602.03218, pp. 1–10, 2016. [Online]. Available: http://arxiv.org/abs/1602.03218

[29] J. Beran, *Statistics for Long-Memory Processes*, vol. 61. Boca Raton, FL, USA: CRC Press, 1994.

[30] D. B. Percival, "Statistics for long-memory processes," *J. Amer. Statist. Assoc.*, vol. 91, no. 435, pp. 1379–1381, 1996.

[31] G. Samorodnitsky, "Long range dependence," *Found. Trends Stochastic Syst.*, vol. 1, no. 3, pp. 163–257, 2006.

[32] P. Doukhan, G. Oppenheim, and M. S. Taqqu, *Theory and Applications of Long-Range Dependence*. New York, NY, USA: Springer, 2003.



ZHIBIN QUAN was born in Hengyang, China, in 1986. He received the B.S. degree in computer science and technology from Jishou University, Jishou, China, in 2009, and the M.S. degree in computer software and theory from Southeast University, Nanjing, China, in 2011.

He is currently pursuing the Ph.D. degree with Southeast University, Nanjing. His current research interests include machine learning and natural language processing.

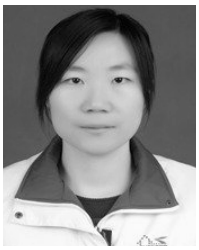
ZHIQIANG GAO was born in Langfang, Hebei, China, in 1966. He received the B.A. and M.D. degrees from Southeast University and the Ph.D. degree from the Department of Mechanical Engineering, Tsinghua University, in 1995. He was with the National CAD Application and Training Network-Nanjing Center and the Department of Mechanical Engineering, Southeast University. In 2000, he held a post-doctoral postion with Department of Social Informatics, Kyoto University, Japan, where he was involved in digital city and artificial intelligence. In 2002, he was an Associate Professor with the Department of Computer Science and Engineering, Southeast University. In 2007, he joined the Institute of Web Science to study ontology learning and natural language processing (NLP), and also large scale reasoning.

His current research interests include material science and manufacturing, numerical simulation of heat transfer and fluid flow, CAD, CG, visualization, VR, game theory, agent oriented programming, Semantic Web, knowledge capture, web mining, military simulation, philosophy, traveling, basketball, and Taijiquan. He was concentrated in human-like and conversational agent, multi-agent system, agent oriented programming, and machine learning. A project was launched in 2003 about interactive multi-agent system used for shooter training simulation, in which agents are hosted by augmented reality. He has to construct 3-D buildings and terrains, design and debug scenarios, and recognize behavior of shooters in the real world. Although this project is more complicated than designing ordinary CG games, it is much more interesting and challenging. He is also involved in 973 Project for semantic grid. Recently, He is responsible for two National Science Foundation of China projects, which are related to ontology learning and inductive logic programming.

Speech recognition, text understanding and image recognition are faced with the same puzzles, and natural language processing is one of the best test beds for various innovative AI ideas. Machine Learning may play a key role in these fields in the future. In addition, semantic annotation of 3-D environments for multi-agents to plan and act "rationally" is an important branch of Semantic Web. He firmly believes that a new era for search engine based on deep NLP, machine learning, and large scale reasoning is coming.

WEILI ZENG received the B.S. degree in applied mathematics from Jishou University, Jishou, China, in 2007, and the M.S. degree in applied mathematics and the Ph. D degree in intelligent transportation systems from Southeast University, Nanjing, China, in 2009 and 2013, respectively.

He is currently an Associate Professor and a Master's Supervisor with the College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing. His current research interests include intelligent transportation systems, computation vision, and machine learning.



XUELIAN LI received the B.S. degree in information and computing science and the M.S. degree in applied linguistics from Nanjing Tech University, Nanjing, China, in 2010 and 2014, respectively.

She is currently pursuing the Ph.D. degree in computer soft science and technology with Southeast University, Nanjing. Her research interests include information extraction, natural language processing, and artificial intelligence.

MAN ZHU received the B.S. degree from the Software College, Jilin University, China, in 2006, the M.S. degree from Hohai University, China, in 2009, and the Ph.D. degree from Southeast University, China, in 2015.

Since 2016, she has been with the School of Computer Science, Nanjing University of Posts and Telecommunications, China. Her research interests include information extraction, medicare fraud detection, statistical machine learning, and Semantic Web.