Computer Science and Application 计算机科学与应用, 2018, 8(1), 49-58
Published Online January 2018 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2018.81008

Hans汉斯

Long-Term Memory Neural Turing Machines

Xiao Xie, Youqun Shi

College of Computer Science Technology, Donghua University, Shanghai
Email: harry_xie@live.com, yqshi@dhu.edu.cn

Received: Jan. 3rd, 2018; accepted: Jan. 22nd, 2018; published: Jan. 29th, 2018

Abstract

The readable and writable external memory module can improve the ability of the neural network which is based on factual memory and memory-based reasoning. Neural Turing Machines use attention mechanism to design a reading and writing mechanism for the memory module. It also realized sorting, copying and other algorithms by using a recurrent neural network as a controller. To shorten the training time and speed up convergence in a wider range of applications (such as natural language processing), we design a kind of read-write mechanism based on the global memory which is applied in Turing machines. This kind of read-write mechanism does convolution operations to extract the global memory features, whose training speed is 6 times better than Neural Turing Machines; the convergence rate and reasoning results in the bAbi dataset are also better.

Keywords

Neural Turing Machines, Long-Term Memory, Working Memory, Random Access, Neural Language Processing Reasoning

长期记忆神经图灵机

解笑，史有群

东华大学，上海
Email: harry_xie@live.com, yqshi@dhu.edu.cn

收稿日期：2018年1月3日；录用日期：2018年1月22日；发布日期：2018年1月29日

摘　要

可读写的外部记忆模块可以在事实的记忆和基于记忆的推理上扩充神经网络的能力。神经图灵机利用注意力机制设计了一种对内存模块的读写机制，并以循环神经网络作为控制器，实现了排序、复制等算法。为了在更广泛的应用(例如自然语言处理)中缩短训练时间或加快收敛速度，我们在神经图灵机的基础上设计

文章引用: 解笑, 史有群. 长期记忆神经图灵机[J]. 计算机科学与应用, 2018, 8(1): 49-58.
DOI: 10.12677/csa.2018.81008

了一种基于全局内存的读写机制，利用卷积运算提取全局内存特征。对于一些较长的序列任务，训练速度相对于神经图灵机提高了6倍，收敛速度也有所提升，在bAbi数据集中取得了更好的推理分类结果。

关键词

神经图灵机，长期记忆，工作记忆，随机读写，自然语言推理

1. 引言

人类所获取的知识种类繁多，有些是能够清晰表达的，例如人类是一种哺乳动物，有些信息是隐含的、难以用语言表达的，例如描述人类的心理状态。神经网络依靠其多层的网络结构能够从简单到复杂表示不同的概念，擅长存储隐性知识，所以神经网络在模式识别任务中可以取得很好的效果[1] [2]。然而神经网络却难以记住非常明确具体的事实[3]，例如在输入序列中陈述了“开会时间是三点半”，而输出结果依赖于这一事实时，神经网络通常无法给出满意的结果。尽管可以对输入做出直观的反应，但对于现代人工智能来说，推理能力是至关重要的[4]。

很多人工智能问题可以归纳为序列问题，循环神经网络(Recurrent Neural Network, RNN)过去在机器翻译、文本生成等序列任务上取得了非凡的成就，得益于其能够在时间序列中学习到如何转换和表示输入数据。但是标准的 RNN 会遇到梯度消失的长期依赖问题，长短期记忆网络(Long Short-term Memory, LSTM)通过学习控制输入和隐藏状态，使反向传播的梯度可以在更长时间序列中进行，从而避免了这一问题[5]。另外，由于 RNN 使用网络结构本身存储过去时刻的内容，仍会遇到无法精确地存储具体事实的问题，对于某些需要记录和推理的任务，改进过的 LSTM 网络依然无法取得好的结果，主要原因是被编码在隐藏状态和权值参数中的事实记忆过于微弱。

解决这一问题的方法是为神经网络添加额外的内存模块，类似人类为实现一些目标而明确保存和操作相关信息片段的行为，将神经网络作为控制器控制内存模块中的写入位置，在需要的时候取出对应位置的信息，从而在少量训练样本中提取更关键的推理信息[6]。Weston 等人因此提出了记忆神经网络[7] [8]，描述了拥有内存模块的神经网络的基本结构，Graves 等人[9]利用 RNN 的图灵完备性[10]提出了神经图灵机(Neural Turing Machine, NTM)，仿照冯诺依曼的计算机结构，使用外部内存模块扩展了 RNN，使其能够完成对输入数据的复制和排序等功能。与神经图灵机的注意力机制不同，神经随机读取机[11]采用强化学习的训练方法，在离散的内存模块上进行训练，实现了一些简易的算法。可微分神经计算机(Differentiable Neural Computer, DNC) [12]通过学习相互关联的信息，实现了伦敦地铁路径搜寻、族谱推断等包含复杂推理功能的任务。

由于这些工作将所有的输入数据均视为有效的重要信息，但是并非所有信息都是重要的，例如在自然语言处理中一些名词和动词比介词和副词更加重要。在这种情况下，神经图灵机会遇到收敛速度过慢、训练时间过长的问题。我们在神经图灵机的基础上提出了基于全局内存的读写机制，使控制器网络在任意时刻可以提取过去多个时刻的信息，这对于大部分序列任务来说能够在有限的训练时间内快速收敛。由于网络结构适用于较长的序列，将其命名为长期记忆神经图灵机(Long-term Memory Neural Turing Machine, L-NTM)。

2. 相关工作

2.1. 神经图灵机

神经图灵机模拟冯诺依曼计算机体系，采用外部存储矩阵作为“内存”，使用 RNN 作为控制器，使用注意力机制与内存进行读写交互[13] [14]，其各部分均可导，因此整个网络可以通过反向传播进行训练。利用 RNN 的图灵完备性，完成了复制、排序等图灵机具备而当前其他神经网络不具备的功能。神经图灵机首次利用外部存储扩展了 RNN 的功能，使神经网络具备了“记住”短期事实的能力，为推理功能提供了记忆基础。

图灵机作为一种计算模型，可以理解为“建立一个输入纸带到输出纸带的映射”，而神经图灵机就是使用神经网络的方法来“建立从输入纸带到输出纸带的映射”以代替人工编写的程序。神经图灵机在复制任务中完全实现了从输入输出数据中学习到了“复制程序”的过程：将输入数据依次存入内存中，输入结束后，循环的读取内存中的数据并将其输出。

2.2. 可微分神经计算机

作为神经图灵机的升级版，可微分神经计算机(DNC)通过动态分配内存、相关记忆的互相连接等改进，极大地增强了神经图灵机的推理能力。相对于 NTM 的只能从相邻位置读取信息，DNC 能够对内存进行随机读写，且能将相关联的信息使用临时连接关联起来，例如在族谱推理中，父亲和子女会用一些临时连接相互关联，在问到有关父亲的问题时，这些临时连接就会起到非常重要的作用。除此之外，DNC 还能够实现更加复杂的推理任务，例如伦敦地铁路线的查询，在描述了伦敦地铁的站点和线路之后，就可以让 DNC 回答例如“从 Bond 街出发，顺着 Central 线沿某方向走一站路，然后沿 Circle 线按某方向走 4 站，再沿 Jubilee 线按某方向走 2 站，最后你会到达哪一站？”

DNC 将神经图灵机的推理能力提高到了更高的层次，同时复杂的结构设计以及对输入信息的处理上使 DNC 在对待普通任务时同样会受到训练过长、收敛速度慢的问题。

2.3. 记忆神经网络

记忆神经网络以问答系统中的推理机制作为出发点，定义了用可读写的内存模块扩充神经网络的通用结构，包括输入到内部表示的映射、内存更新机制以及内部表示到输出的映射。Weston 的《记忆神经网络》更加关注自然语言处理的应用，文中讨论了在应对大文本的内存写入机制，以及当测试集中出现新词时如何应对的问题，同时设计了一些对于非重要信息的遗忘机制，我们在 L-NTM 中也应用了这一机制。

3. 模型结构

3.1. 基础结构

作为神经图灵机的一个分支，L-NTM 的结构如图 1 所示，包括一个可供读写的记忆存储模块或称为内存模块 $\boldsymbol{S}$ 和一个神经网络控制器。 $\boldsymbol{S}$ 是一个 $M \times N$ 的矩阵，用于存储需要的数据，是记忆的载体，M 为内存模块的长度，表示存储数据量的上限，N 为每一条存储的维度。

控制器在每一个时刻 t 都会对 $\boldsymbol{S}$ 进行读取和写入，它的学习目标就是在不同时刻将不同的输入存储到 $\boldsymbol{S}_t$ 中的不同位置，并读取相关的记忆用于输出。为了确保能够使用梯度下降进行端到端的训练，需要在整个计算过程中每一个部分都是可微的，因此控制器输出的地址指针不能指向具体的每某个位置，而是输出一个长度为 M 的指针向量 $\boldsymbol{w}_t$，其定义与 NTM 一致：

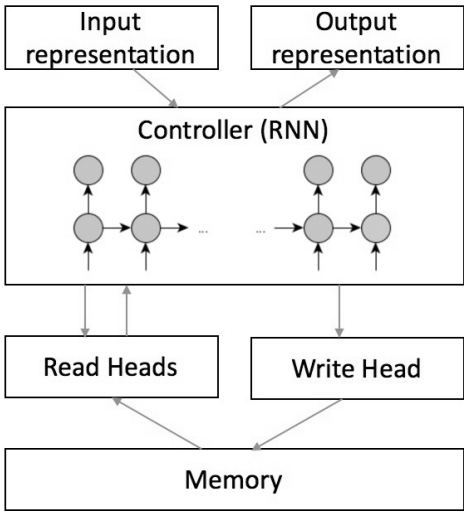


Figure 1. Basic structure of neural Turing machines
图 1. 神经图灵机基本结构

$$\sum_k w_t(k)=1,\ 0\le w_t(k)\le 1,\ \forall k \qquad \#(1)$$

向量中每一个元素代表在该位置写入的数据量。例如当 $N=3$，$w_t(k)$ 为 $[0,1,0,1,0.8]$ 时，控制器会主要在第三个位置写入信息。

L-NTM 的控制器中的核心网络采用 LSTM 网络，LSTM 有更好的能力应对序列中的长期依赖问题，从而学习到更好的与外部内存交互的功能。

3.2. 改进的读写方法

通常网络每一个时刻的输出，可能会依赖于多个前期输入，因此在 L-NTM 中，控制器将会生成多个指针，用于读取多个前期数据。

定义在时刻 t 第 i 个读取指针为 $\boldsymbol{w}_t(i)$，所读取的内容为长度为 N 的行向量 $\boldsymbol{r}_t(i)$，其内容为 $\boldsymbol{S}_t$ 中每一行对应读取指针的加权叠加：

$$\boldsymbol{r}_t(i)\leftarrow\sum_j \boldsymbol{w}_t(i,j)\boldsymbol{S}_t(j) \qquad \#(2)$$

生成读取向量后，控制器中的输出映射函数 $\boldsymbol{O}$ 会将 L-NTM 的内部表示映射到具体的输出：

$$\boldsymbol{o}_t=\boldsymbol{O}(\boldsymbol{r}_t) \qquad \#(3)$$

写入时，由于在每个时刻只有单个输入，所以只需一个写入指针 $\boldsymbol{w}_t$，整个写入过程分为两个部分：擦除[7]和写入。在时刻 t，控制器生成擦除向量 $\boldsymbol{e}_t$、写入向量 $\boldsymbol{a}_t$ 和控制写入量的参数 f_t。其中 $\boldsymbol{e}_t$ 和 $\boldsymbol{a}_t$ 均为 N 维向量，且 $\boldsymbol{e}_t$ 中所有元素和参数 f_t 的数值范围均在(0, 1)内。令经过擦除操作的内存为 $\tilde{\boldsymbol{S}}_t$：

$$\tilde{\boldsymbol{S}}_t(i)\leftarrow\boldsymbol{S}_{t-1}(i)\left[\boldsymbol{1}-\boldsymbol{w}_t(i)\boldsymbol{e}_t\right] \qquad \#(4)$$

当 $\boldsymbol{e}_t$ 中的元素越接近于 0，则表示上一时刻在对应位置的内容会被保留，越接近于 1，这部分内容会越接近于被清除。擦除操作之后会执行写入操作，$\boldsymbol{a}_t$ 为在t时刻将要写入内存的信息，类似于 LSTM 中的输入门，此处会有一个控制参数 f_t 决定当前输入在内存中的存储量。

在一些自然语言处理的任务中，人们的语言中包含了很多对于记忆没有必要的介词等词语，过滤掉

这些非关键词是非常重要的。

$$\boldsymbol{S}_t \leftarrow \tilde{\boldsymbol{S}}_t(i) + f_t \boldsymbol{w}_t(i)\boldsymbol{a}_t \quad \#(5)$$

需要注意这里读取指针和写入指针是不相关的，控制器会分别生成两个对应的指针。

3.3. 改进的写入寻址机制

与 NTM 对读写采用同一种寻址机制不同，L-NTM 分别对读写采用了不同的寻址机制。对于写入指针，受 DNC 的启发，在确定写入位置时，应考虑整体内存的使用情况和记忆的存储分布。卷积运算提供了一种基于局部并可在整体上进行特征提取的方法[15]，因此首先使用二维相同(same)卷积核 K 生成一个长度为 N 的向量 $\boldsymbol{w}_t^l$，其中 K 的输入通道数为 N，设卷积核大小为 k。

$$w_t^l(i,j) = (\boldsymbol{S}_t * K) = \sum_m \sum_n \boldsymbol{S}_t(i+m, j+n) K(m,n) \quad \#(6)$$

其中，$m \in [-k,k], n \in [0, N-1]$,且 $m,n \in \mathbb{Z}$。

k 的大小决定了对内存模块局部特征提取的范围，k 越大，内存中对应行的特征会关注更大的局部区域，同时边缘内存对全局的影响会降低。在我们的实验中内存长度比较短，因此更需要关注局部的内存特征，k 的取值不需要过大，在实验中均取为 3。

由卷积核提取的特征向量 $\boldsymbol{w}_t^l$ 的元素有正有负，不符合指针向量的要求(所有元素均大于等于 0，且和为 1)，因此我们采用 softmax 函数令每一个元素符合指针要求：

$$\tilde{\boldsymbol{w}}_t = \text{softmax}\left(\boldsymbol{w}_t^l\right) \quad \#(7)$$

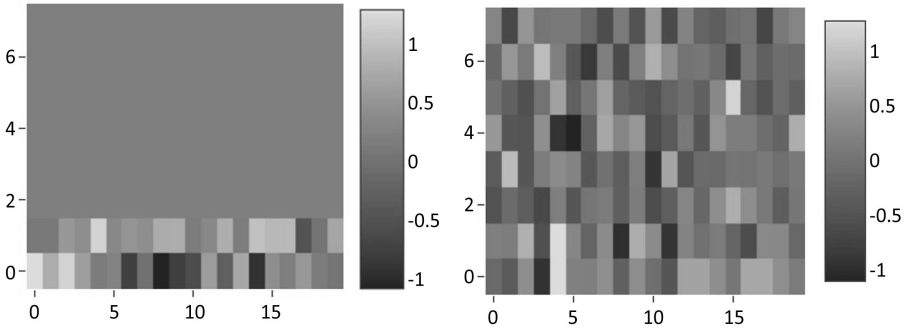
如果内存模块中的存储比较分散，例如[[0.2, 0.3], [0.1, 0.4], [0.4, 0.3]]，而不是[[0.9, 0.8], [0.1, 0.4], [0.2, 0.3]]这种结构非常清晰的存储模式，这会导致 $\boldsymbol{w}_t$ 中的位置过于分散，例如[0.3, 0.3, 0.4]，而不是[0.8, 0.1, 0.1]。分散的存储在规模较小的内存结构中不会存在太大问题，但当内存的长度即 N 较大时，相对于有限数量的读取指针，分散的存储不利于信息的读取。因此控制器会生成一个大于 1 的参数 γ_t 用于写入指针的聚焦：

$$w_t(i) \leftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}} \quad \#(8)$$

L-NTM 与 NTM 都有指针的聚焦操作，L-NTM 的目的是避免在规模较大的内存中的写入位置过于分散，这会导致读取指针数量不是足够大的时候无法获取到大部分有效信息，而 NTM 的聚焦更倾向于聚焦到某一行中，对于同样的输入，NTM 会出现[0.05, 0.9, 0.05]这样的指针，而[0.25, 0.5, 0.25]对于 L-NTM 就已经足够。由于聚焦程度的不同，经过一段时间读写的内存状态也会有所不同。如图 2(a)所示，在 NTM 中，会更倾向于在某一个位置写下信息，此时的内存状态按倒序依次在内存中进行写入，在其他地方的数据量则会很小，图 2(b)所示为 L-NTM 的内存状态，聚焦程度较小时，写入的信息量会比较分散。

在 NTM 中，内存模块的初始化全部为接近于 0 的非常小的数，例如10^{-6}，这会导致卷积运算的结果 $\boldsymbol{w}_t^l$ 除了边界元素都相等，这会对写入指针的聚焦操作造成很大的困难，因此在 L-NTM 使用高斯分布对内存模块进行初始化，不同位置的存储越不相同，卷积运算后提取的特征越不一致，这将使最终的写入指针更加聚焦与某一位置。

完整的流程如图 3 所示，首先利用卷积运算提取出内存的特征，之后利用 softmax 函数令向量满足指针形式，最终利用聚焦运算将写入位置聚集到某几个位置上。



(a) NTM 内存示例　　(b) L-NTM 内存示例

Figure 2. Example of memory
图 2. 内存示例，(a)为 NTM 的内存示例，在 NTM 中，内存会偏向于按顺序读写，而在 L-NTM 中，则更加偏向于全局读写

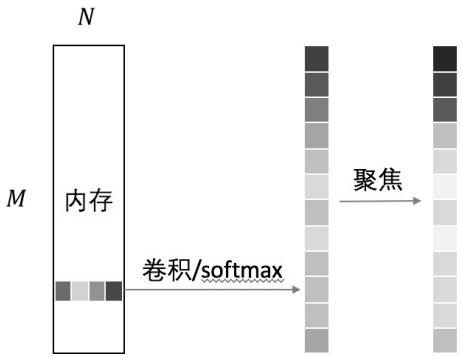


Figure 3. Addressing mechanisms of writing heads
图 3. 写入指针的寻址机制

3.4. 改进的读取寻址机制

生成读取指针的过程分为内容寻址和全局寻址两个过程。内容寻址与 NTM 一致，控制器生成一个 N 维行向量 $\boldsymbol{c}_t$，利用相似度度量方法 $\mathrm{K}[\cdot,\cdot]$ 与 $\boldsymbol{S}_t$ 中每一行 $\boldsymbol{S}_t(i)$ 进行比较，生成基于内容的读取指针 $\boldsymbol{w}_t^c$：

$$w_t^c(i) \leftarrow \frac{\exp\left(\beta_t \mathrm{K}\left[\boldsymbol{c}_t, \boldsymbol{S}_t(i)\right]\right)}{\sum_j \exp\left(\beta_t \mathrm{K}\left[\boldsymbol{c}_t, \boldsymbol{S}_t(j)\right]\right)} \quad \#(9)$$

其中 β_t 是一个用于增强或减弱这一时刻的基于内容的寻址权重。内容寻址的目的是在该时刻想要获取什么样的记忆存储，与精确地取得记忆不同，获取大致的想要的内容不需要大量的数据用于训练，同时又足够灵活能够应付更多种类的输入情况。

全局寻址采用的方法同样使用了卷积运算，与写入机制寻址的方法不同的是，在读取的全局寻址中使用多个卷积核，这是为了能够更好的分析内存模块以多种特征向量，同时针对不同的特征向量 $\boldsymbol{w}_t^l(i)$，生成不同的控制参数 $g_t(i)$，$g_t(i)$ 的范围在(0,1)内，用于将 $\boldsymbol{w}_t^c$ 与 $\boldsymbol{w}_t^l(i)$ 结合起来：

$$\boldsymbol{w}_t(i) \leftarrow g_t(i)\boldsymbol{w}_t^c + \left(1 - g_t(i)\right)\boldsymbol{w}_t^l(i) \quad \#(10)$$

上述所有的操作均为可微的，因此 L-NTM 可以使用梯度下降进行端到端的训练。由于中间有大量的卷积运算，得益于卷积运算可以并行计算的特性，L-NTM 的训练和运行都会以更快的速度执行。图 4 展示了读取指针的寻址机制。

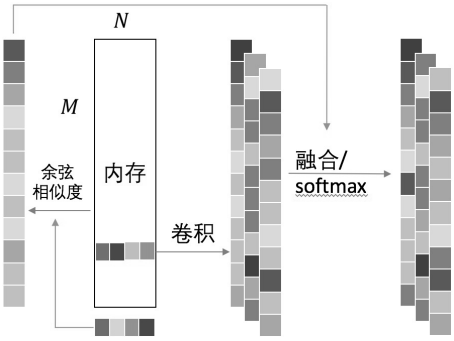


Figure 4. Addressing mechanisms of reading heads
图 4. 读取指针的寻址机制

从强化学习[16] [17]的角度来看，将内存模块视为状态(state)，那么写入机制是对状态转换的函数进行建模，读取机制则为依据当前状态进行动作(action)选择[18]。通过端到端的学习训练，读取机制会逐渐学习到不同内存状态下的读取位置，而写入机制则会学习到依据上一时刻的内存状态判断当前的写入位置。一旦某一个机制趋于稳定，那么后续训练则会按照类似于强化学习的训练过程不断优化以达到最优结果。

4. 实验

bAbi 数据集[19]是包含一系列推理任务的问答数据集，数据形式为故事陈述、问题和答案组成。我们使用了其中“Single Supporting Fact”和“Two Supporting Facts”两个子数据集用于 NTM 和 L-NTM 的比较，在“Single Supporting Fact”任务中，答案仅与陈述中的一句话有关联，只需一个事实支撑即可得到正确答案，在“Two Supporting Facts”中，为获取答案需要两句陈述支持，并且需要一定的推理才能完成。形式如图 5 所示。

对于问答形式的任务，通常会使用两种处理方法[19] [20]：1) 序列到序列的方法；2) 陈述语句和问题分为两部分模块；如图 6 所示。在这里采用第一种方法，相比于第二种方法，第一种方法对应更长的序列和更加通用的序列任务，这可以在一定程度上检验模型的泛化能力。

我们使用 LSTM 和 NTM 作为基线，整个数据集包括 10,000 条训练数据和 1000 条测试数据，使用 PyTorch 实现模型并在一块 NVIDIA P100 的 GPU 上进行训练。

内存参数设为$M = 8，N = 20$来初始化 NTM 和 L-NTM 的内存结构，在 L-NTM 中，分别设置了读取指针书量为 8、16 和 32，从图 7 中可以看出，NTM 的收敛速度在 L-NTM 的 8 个读取指针和 16 个读取指针之间，小于 32 个读取指针的收敛速度。

读取指针的数量越多，所能提取内存特征也越多，误差的下降速度也会更快，同时由于参数的增多，所采取的正则化项也更严格。实验结果见表 1，括号中的时间为一次迭代所需的时间。

在训练速度上 L-NTM 相对于 NTM 来说提高了 6 倍，得益于 GPU 的并行计算，读取指针的数量增加几乎不会增加训练时间，因此读取指针分别为 8、16 和 32 时，一次迭代的训练时长均为 4 分 40 秒左右。

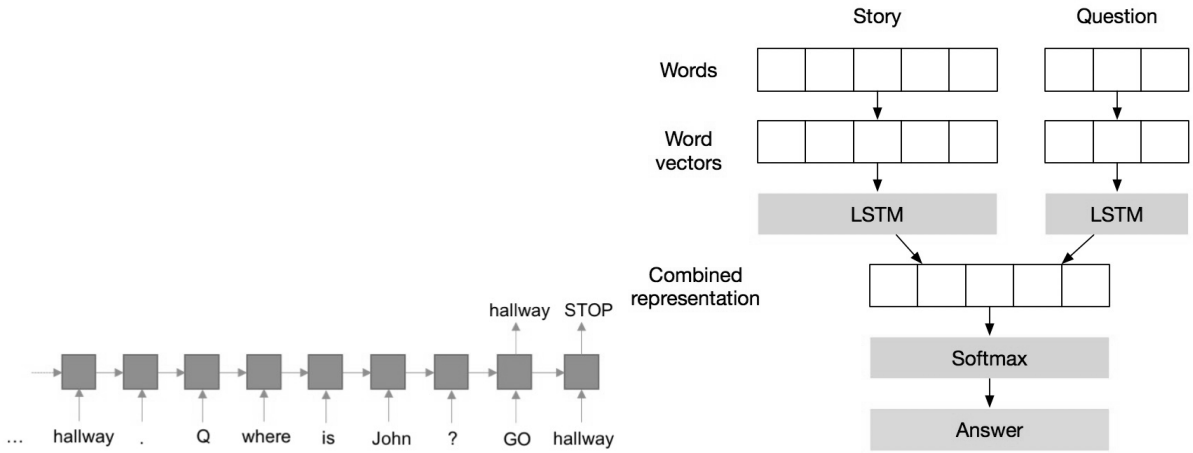
对于 L-NTM 能够取得如此效果的一个可能的原因是大部分序列任务本不需要过于精确地读写，大致的范围就已足够实现目标。由于序列中有很多“to”、“the”等单词，这些无用信息不需要被精确地存储下来，而 NTM 将这些信息都存储下来，浪费了存储空间和读取的时间。

接下来仅使用 LSTM 作为基线与 L-NTM 在“yes/no questions”、“simple negation”、“basic deduction”和“agent’s motivation”任务中进行比较，实验结果为表 2。表明了 L-NTM 在推理任务中的表现要优于

Single Supporting Fact：
1. John travelled to the hallway.
2. Mary journeyed to the bathroom.
3. Where is John? hallway 1

Two Supporting Facts：
1. Mary got the milk there.
2. John moved to the bedroom.
3. Sandra went back to the kitchen.
4. Mary travelled to the hallway.
5. Where is the milk? hallway 1 4

Figure 5. Examples of datasets
图 5. 数据集示例



(a) 序列到序列的模型　　(b) 陈述与问题分开式的模型

Figure 6. Structure of question answering models
图 6. 问答模型结构，(a) 采用序列化方法，使用特殊符号作为故事陈述和问题的分隔符，在特殊符号“Q”后为问题序列，特殊符号“GO”之后的输出为答案；(b) 将故事陈述和问题分开为两个模块，分别经过 LSTM 序列化处理最终联合起来共同生成答案

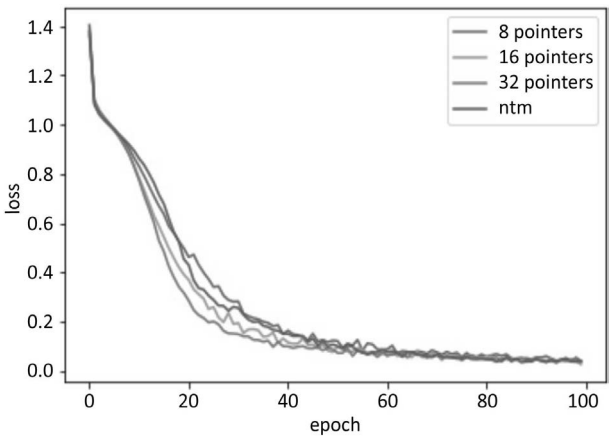


Figure 7. Training results of “Single Supporting Fact”
图 7. “Single Supporting Fact”训练结果

Table 1. Experiment results of LSTM, NTM and L-NTM
表 1. LSTM、NTM 和 L-NTM 的实验结果

数据集	LSTM	NTM	L-NTM
Single Supporting Fact	39.0% (100 s)	100% (25 m)	100% (4 m 40 s)
Two Supporting Facts	20.1% (400 s)	90% (1 h 20 m)	99.6% (18 m)

Table 2. Experiment results of LSTM and L-NTM
表 2. LSTM 和 L-NTM 的实验结果

数据集	LSTM	L-NTM
Yes/no questions	52.5%	100%
Simple negation	64.6%	99.1%
Basic deduction	23.5%	100%
Agent's motivation	83.8%	99.8%

LSTM，这表明了具体的记忆存储可以提高推理任务的准确率。在简单的问答任务中，L-NTM 将陈述的事实内容存储在内存空间中，在问题中序列中通过搜索存储内容进行推理，得到了更好的结果。因此在一些类似的序列任务中，L-NTM 在一定程度上能够替代 LSTM。

5. 结论

以神经图灵机为基础改进了写入方法和读写指针的寻址机制：在写入方法中加入了用于控制写入量的控制参数，一些相对于实体名词含有信息量较少的副词、介词等输入将不会占用过多的内存空间，减少了推理行为中对于无效信息的处理；在寻址机制中，使用卷积运算提取内存模块的多种特征和基于内容寻址的相互结合，在某一时刻可以提取更多的事实依据用于推理；以上改进在 bAbi 数据集的实验中不仅加快了收敛速度，同时也在训练速度上提升了 6 倍，并且在更长的序列中得到了更高的准确率。在与 LSTM 的对比中，L-NTM 体现了更强的推理能力，在所有的推理任务中均取得了最佳结果。

参考文献 (References)

[1] Zeiler, M.D. and Fergus, R. (2014) Visualizing and Understanding Convolutional Networks. *European Conference on Computer Vision*, September 6 2014, 818-833. https://doi.org/10.1007/978-3-319-10590-1_53

[2] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 1097-1105.

[3] Goodfellow, I., Bengio, Y. and Courville, A. (2016) Deep Learning. The MIT Press, Cambridge, 418-422.

[4] Hinton, G.E. (1990) Mapping Part-Whole Hierarchies into Connectionist Networks. *Artificial Intelligence*, **46**, 47-75. https://doi.org/10.1016/0004-3702(90)90004-J

[5] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, **9**, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[6] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D. and Lillicrap, T. (2016) One-Shot Learning with Memory-Augmented Neural Networks. arXiv preprint arXiv: 1605.06065

[7] Weston, J., Chopra, S. and Bordes, A. (2014) Memory Networks. arXiv preprint arXiv:1410.3916

[8] Sukhbaatar, S., Weston, J. and Fergus, R. (2015) End-to-End Memory Networks. *Advances in Neural Information Processing Systems*, 2440-2448.

[9] Graves, A., Wayne, G. and Danihelka, I. (2014) Neural Turing Machines. arXiv preprint arXiv:1410.5401

[10] Siegelmann, H.T. and Sontag, E.D. (1995) On the Computational Power of Neural Nets. *Journal of Computer and System Sciences*, **50**, 132-150. https://doi.org/10.1006/jcss.1995.1013

[11] Kurach, K., Andrychowicz, M. and Sutskever, I. (2015) Neural Random-Access Machines. arXiv preprint arXiv:1511.06392

[12] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J. and Badia, A.P. (2016) Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature*, **538**, 471-476. https://doi.org/10.1038/nature20101

[13] Bahdanau, D., Cho, K. and Bengio, Y. (2014) Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473

[14] Mnih, V., Heess, N. and Graves, A. (2014) Recurrent Models of Visual Attention. *Advances in Neural Information Processing Systems*, 2204-2212.

[15] LeCun, Y. and Bengio, Y. (1995) Convolutional Networks for Images, Speech, and Time Series. *The Handbook of Brain Theory and Neural Networks*, **3361**, 1995.

[16] Li, Y. (2017) Deep Reinforcement Learning: An Overview. arXiv preprint arXiv:1701.07274

[17] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S. (2015) Human-Level Control through Deep Reinforcement Learning. *Nature*, **518**, 529-533. https://doi.org/10.1038/nature14236

[18] Reed, S. and De Freitas, N. (2015) Neural Programmer-Interpreters. arXiv preprint arXiv:1511.06279

[19] Weston, J., Bordes, A., Chopra, S., Rush, A.M., van Merriënboer, B., Joulin, A. and Mikolov, T. (2015) Towards Ai-Complete Question Answering: A Set of Prerequisite Toy Tasks. arXiv preprint arXiv:1502.05698

[20] Stroh, E. and Mathur, P. (2016) Question Answering Using Deep Learning.