法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

第七节课：**DeepMind**的神经图灵机

Neural Turing Machine

Differentiable Neural Machine

Reasoning

本节内容

- Neural Turing Machine
- Differentiable Neural Machine
  - DNC原理
  - DNC 代码演示
- Neural Reasoning最新进展
  - Relational reasoning
  - Recurrent Entity Network

参考文献

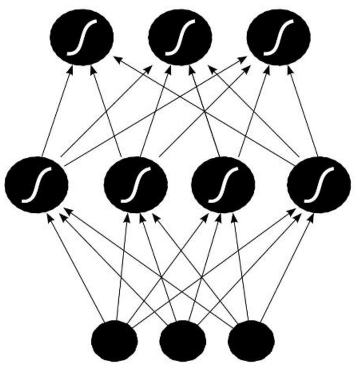
- Neural Turing Machine
  - Neural Turing Machine (2014)
- Differentiable Neural Machine
  - Hybrid computing using a neural network with dynamic external memory (2016)
- DNC 代码演示
  - Implementation and optimization of differentiable neural computers
- Neural Reasoning最新进展
  - A simple neural network module for relational reasoning (2017-06)
  - Tracking the world state with recurrent entity network (2017)

概述

- NTM/DNC 和 Memory Network的异同
  - 都是从model architecture层面，将多个machine learning model联合起来处理复杂的任务
    - CNN：object detection
    - LSTM: sentence representation
    - MemNet/NTM: query and reasoning
  - NTM/DNC花费更多努力在记忆管理上
    - MemNet 注重 memory 查询
    - NTM/DNC 注重更新memory和memory的时间关系
  - Memory Network侧重QA任务，NTM/DNC侧重算法任务
    - NTM/DNC是一个“黑盒子”，自动从数据学习算法
    - **Computer** that learns from examples

NTM

Turn neural networks into differentiable neural computers by giving them read-write access to external memory



'CPU'

+

Memory

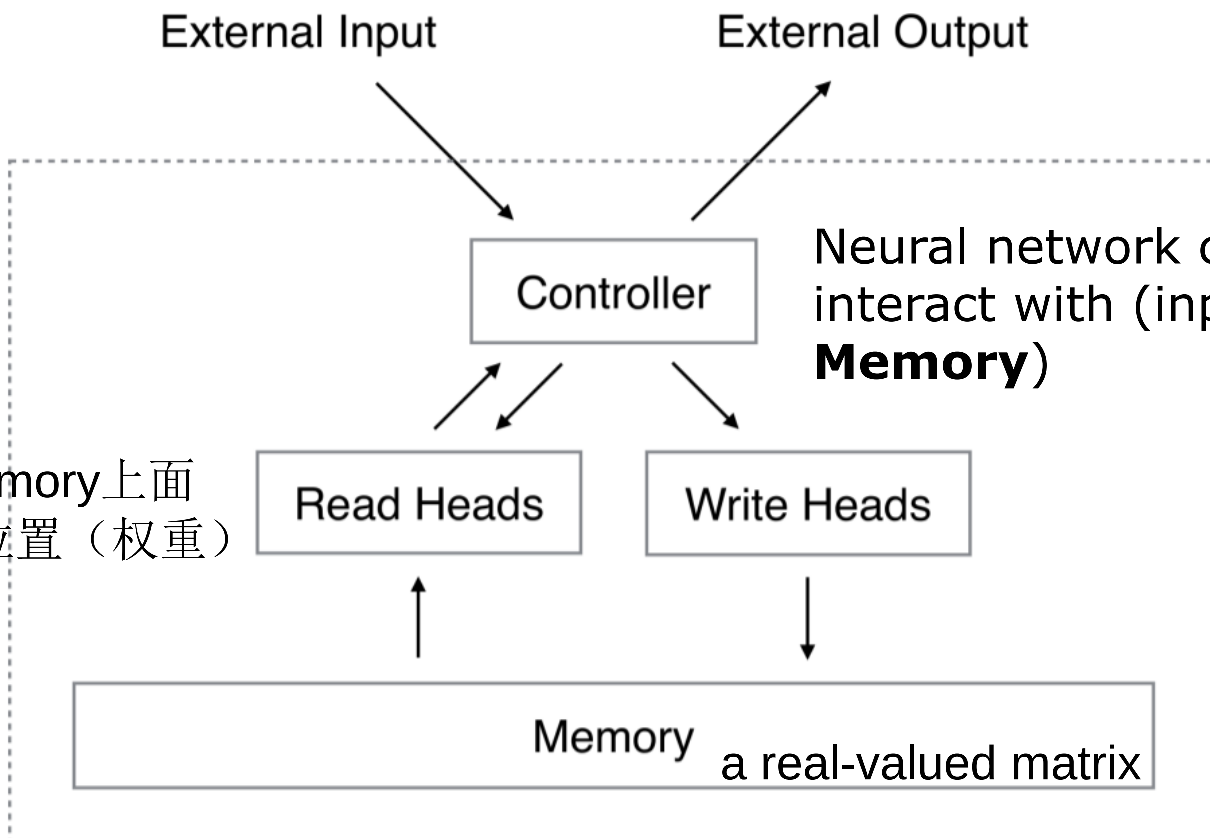
=

Computer that learns from examples

(*or* neural net that separates computation from memory)

图片来源

NTM

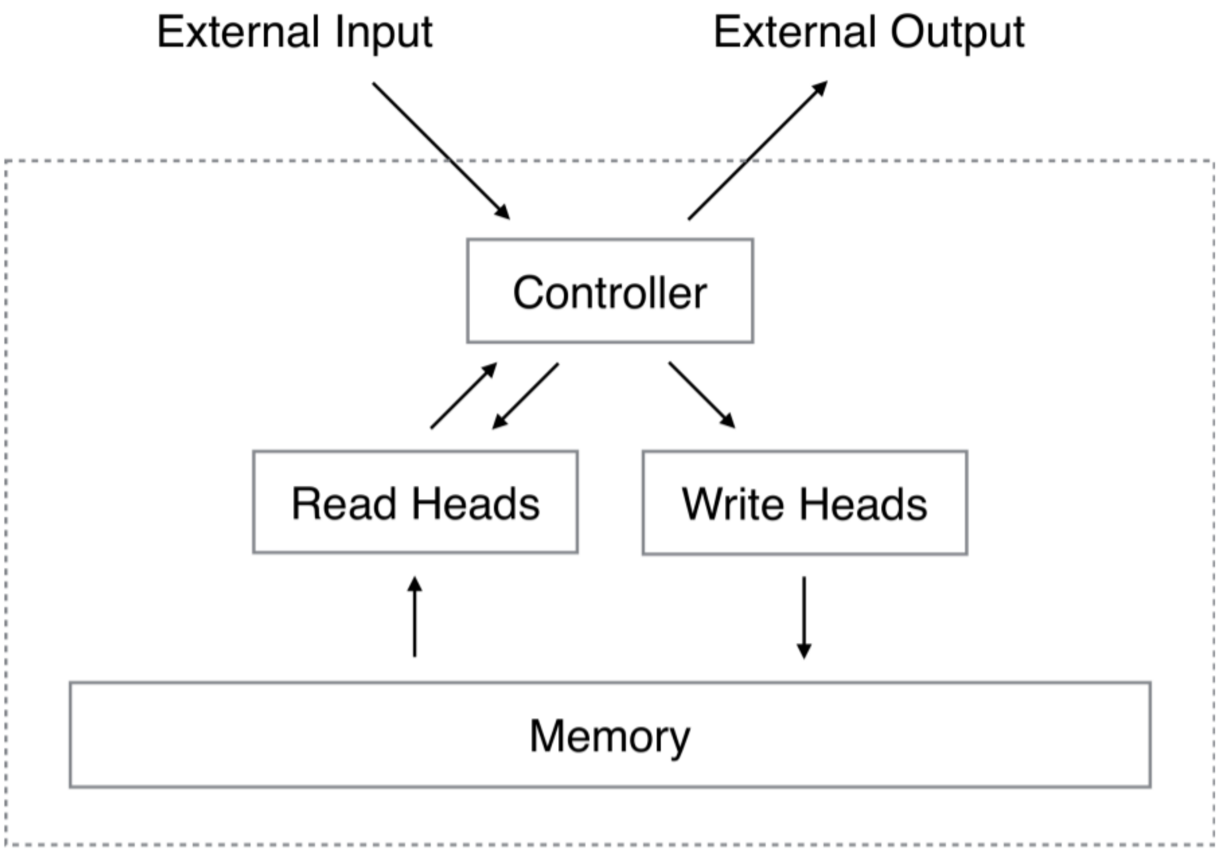


Neural network controller interact with (input, output, **Memory**)

Heads 在Memory上面定位读写的位置（权重）

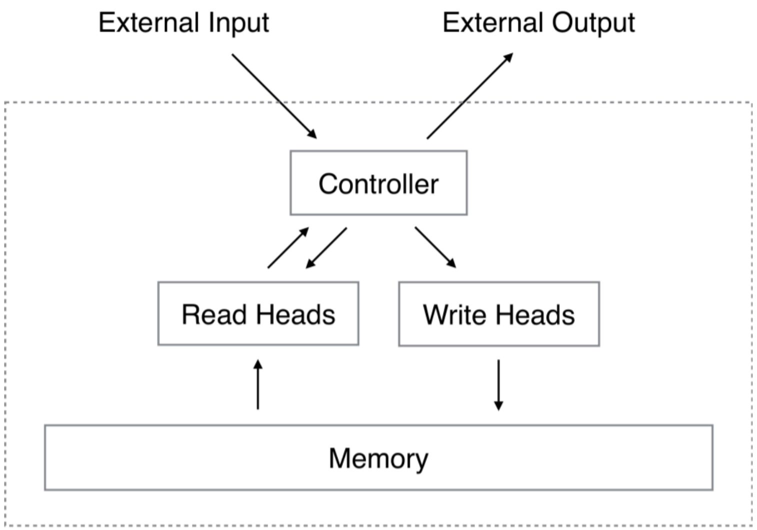
Controller <-- selective read & write -->Memory bank

NTM


External Input
External Output
Controller
Read Heads
Write Heads
Memory

所有操作皆可导

NTM 读操作



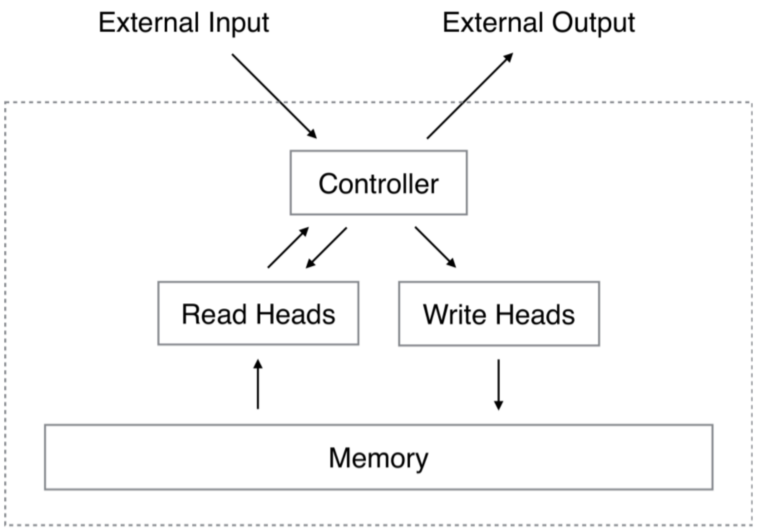
$\mathcal{M}_t$: $N \times M$矩阵，时间t的记忆

N: 记忆向量的数量

M: 记忆向量的维度

- "NTM uses an attentional process to read from memory"
- 使用attention原理计算每个记忆向量的权重
  - $0 < \omega_t(i) \leq 1$
  - $\sum_{i=1}^{R} \omega_t(i) = 1$
- 记忆加权生成read操作的结果
  - $r_t \leftarrow \sum_{i=1}^{N} \omega_t(i) \mathcal{M}_t(i)$

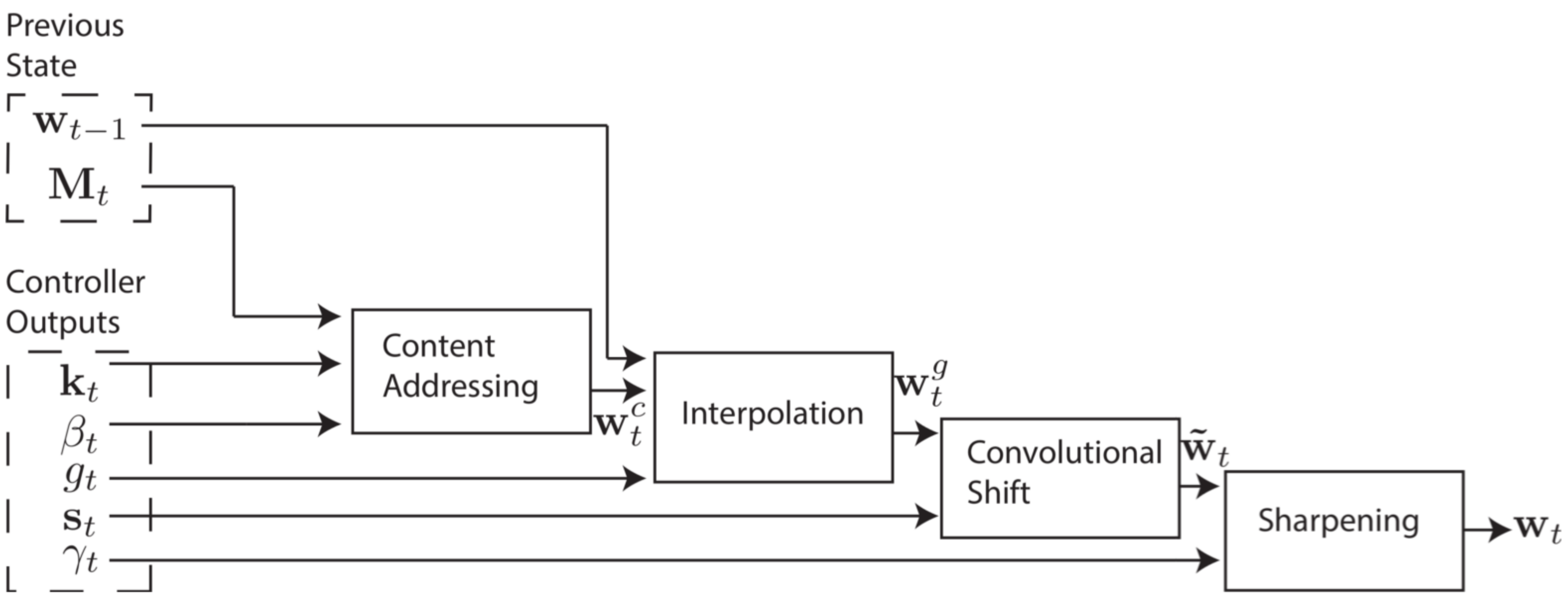
NTM 写操作



$\mathcal{M}_t$: $N \times M$矩阵，时间t的记忆
N: 记忆向量的数量
M: 记忆向量的维度

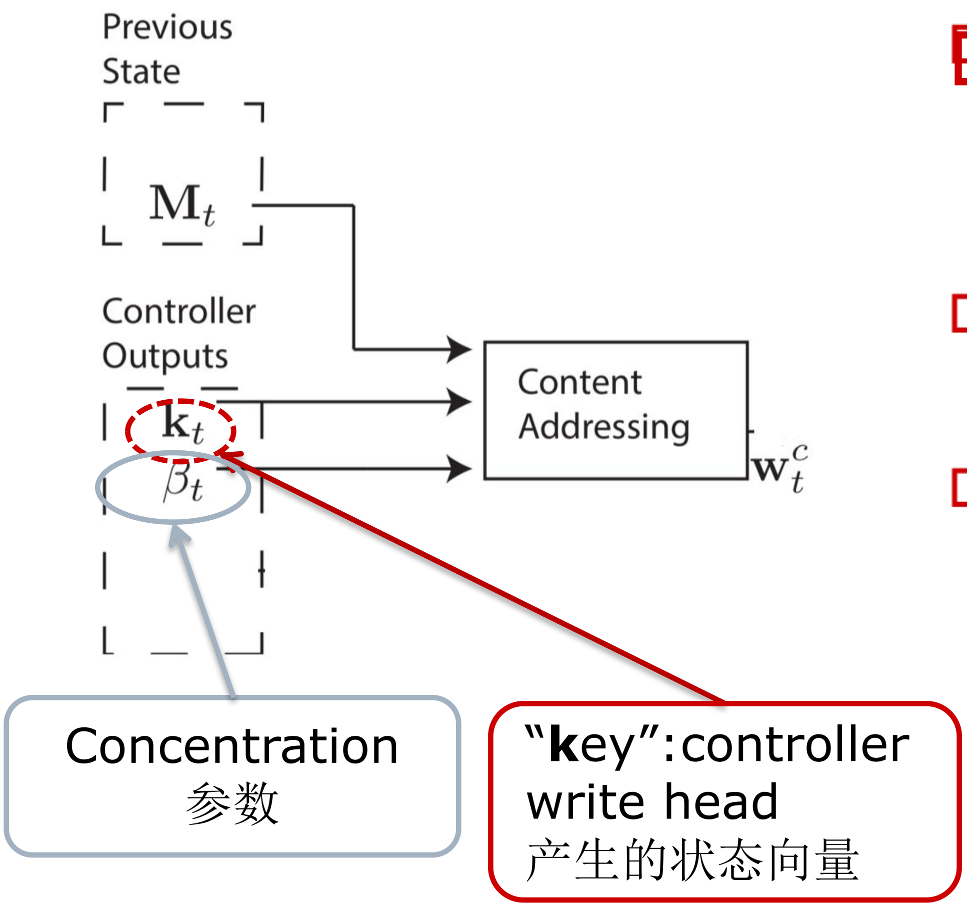
- 对memory bank的写操作包含erase和add两个步骤
- erase操作
  - $e_t \in R^M$: erase向量，$e_t(d) \in (0,1)$
  - $\mathcal{M}_t^{erased}(i) \leftarrow \mathcal{M}_{t-1}(i)[\mathbf{1} - \omega_t(i)e_t]$
- add 操作
  - $a_t \in R^M$: add向量，$a_t(d) \in (0,1)$
  - $\mathcal{M}_t(i) \leftarrow \mathcal{M}_t^{erased}(i) + \omega_t(i)a_t$
- 所有记忆向量共享e_t，a_t

NTM 写操作细节



Addressing mechanism
计算attention权重的流程

NTM addressing 细节 I: Content addressing



- 基于 (1)当前每一个记忆向量和 (2) controller状态向量的相似程度，计算记忆向量的attention权重
- 使用cosine similarity计算相似程度：$K(u,v)=\frac{u\cdot v}{||u||\cdot||v||}$
- 使用softmax 将相似度转化为权重：

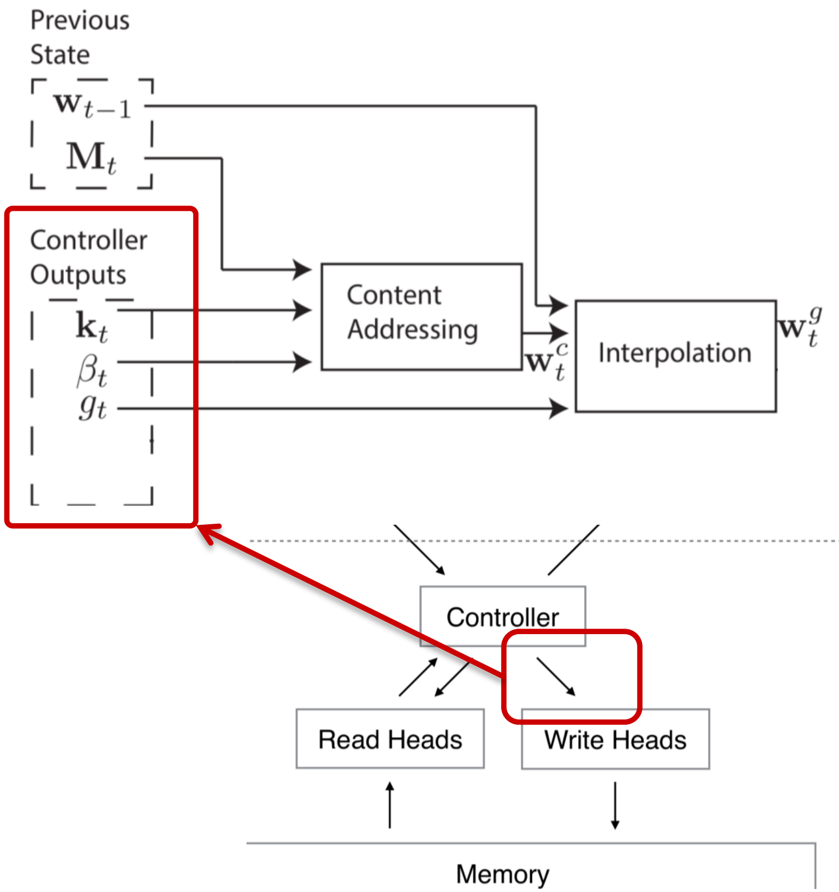
$$\omega_t^c(i) \leftarrow \frac{\exp(\beta_t K[k_t, M_t(i)])}{\sum_j \exp(\beta_t K[k_t, M_t(j)])}$$

NTM addressing细节：Location-based addressing

- QA任务
  - **John is in the playground.**
  - Bob is in the office.
  - **John picked up the football.**
  - Bob went to the kitchen.
  - **Q: Where is the football?**
- 基于内容相似程度从memory bank中寻找相关记忆

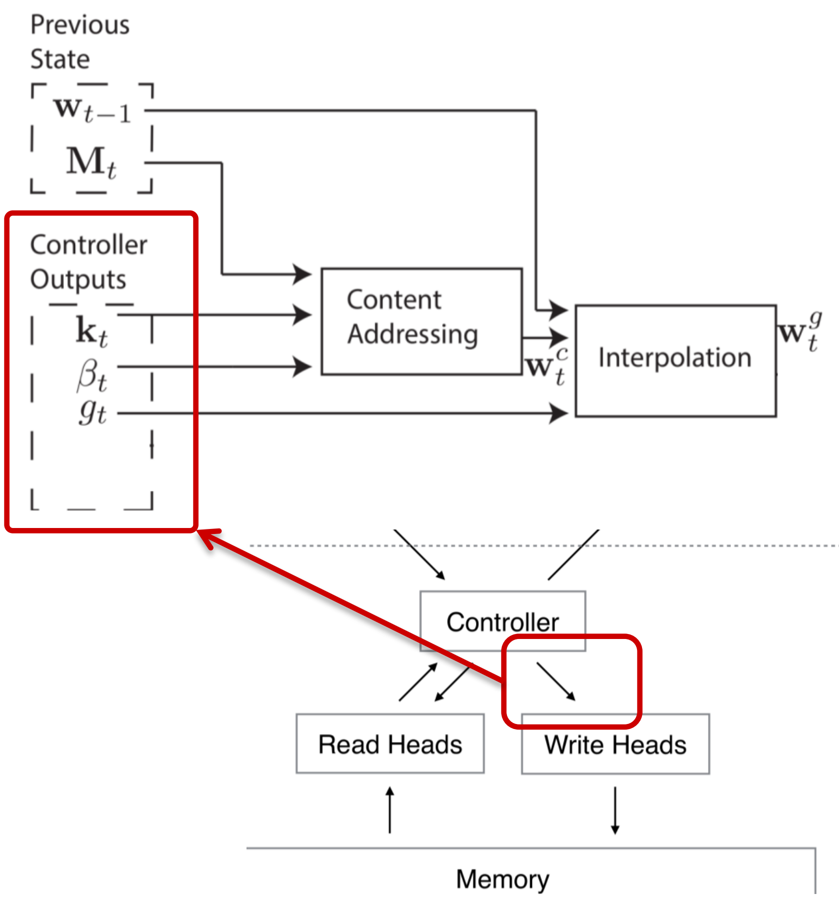
- Algorithmic 任务
  - 计算$f(x, y) = x * y$
  - x, y保存在内存对应的**地址**上
  - 通过地址，而不是通过数值，读写x, y
- Location-based addressing

NTM addressing细节 II: Interpolation Gate



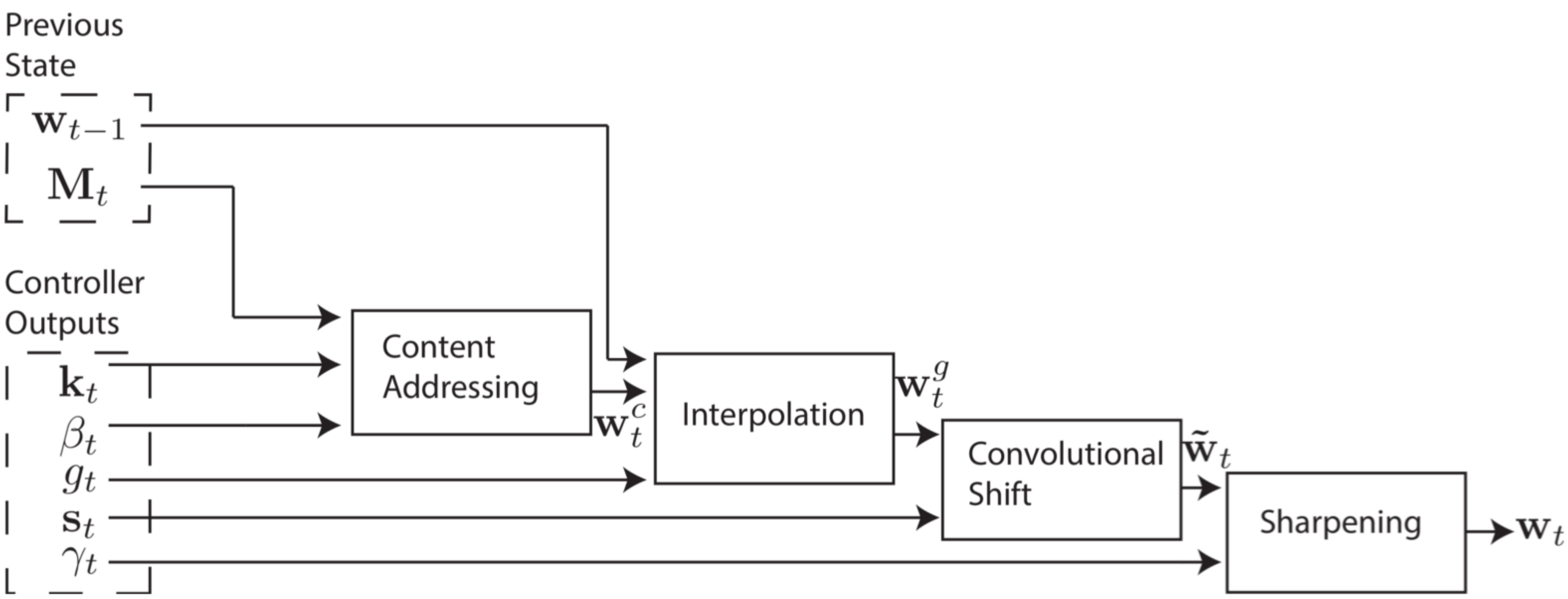
- $\omega_t^g \leftarrow \boldsymbol{g_t}\,\omega_t^c + (1 - g_t)\omega_{t-1}$
- 基于内容的weight vector ω_t^c 和上一个时间的weight vector ω_{t-1}的线性组合
- 线性组合的参数是一个scalar g_t, 由controller预测产生
- Interpolation **gate**决定多大程度上使用content-based addressing

NTM addressing细节 II: Interpolation Gate



- $\omega_t^g \leftarrow \boldsymbol{g_t}\, \omega_t^c + (1 - g_t)\omega_{t-1}$
- 基于内容的weight vector ω_t^c 和上一个时间的weight vector ω_{t-1}的线性组合
- 线性组合的参数是一个scalar g_t, 由controller预测产生
- Interpolation **gate**决定多大程度上使用content-based addressing

NTM addressing细节 III,IV: 后续处理



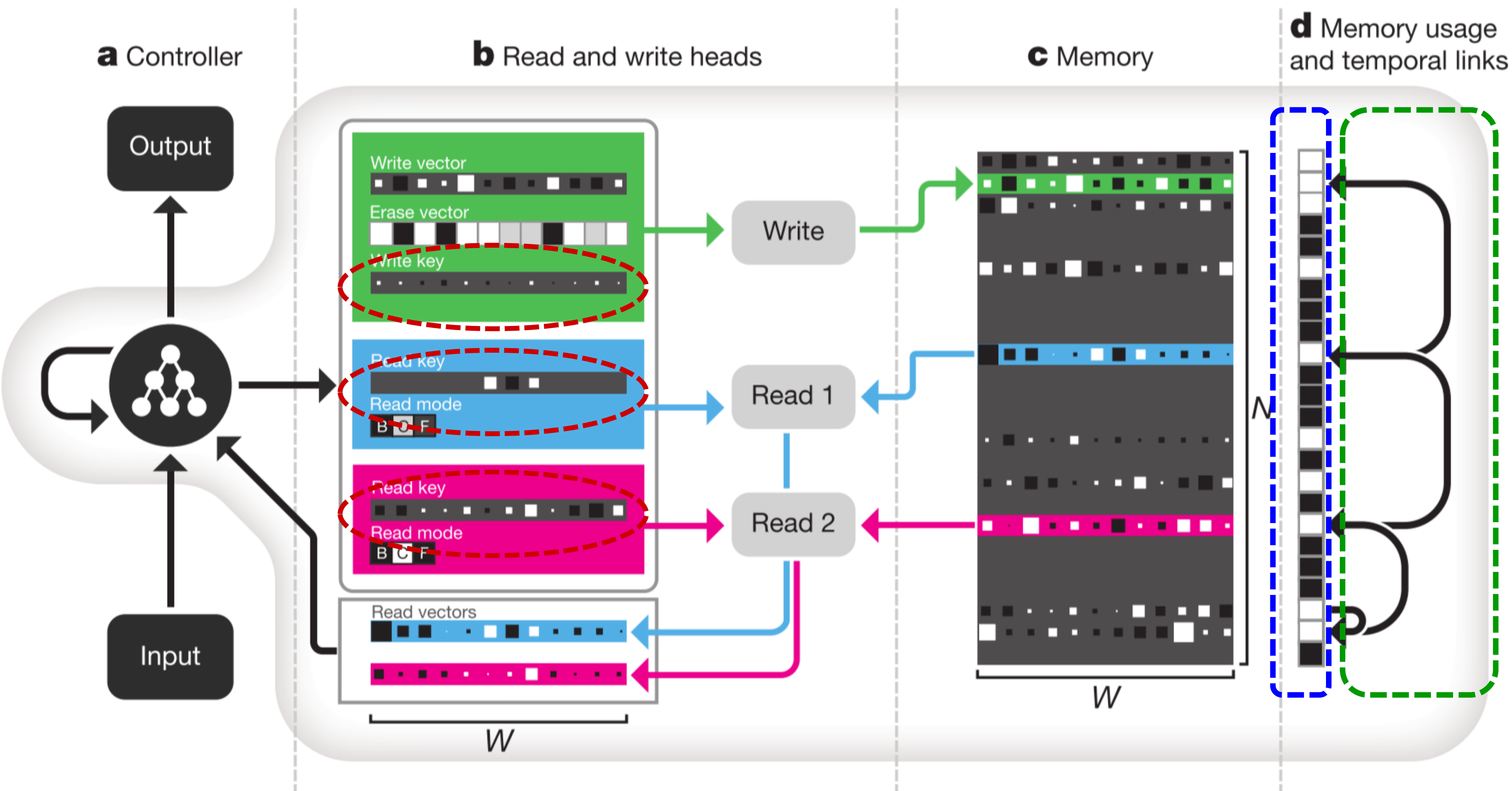
Shift attention: $\widetilde{\omega}_t \leftarrow \sum_{j=0}^{R-1} \omega_t^g(j) s_t(i-j)$

Sharpening: $\omega_t(i) \leftarrow \frac{\widetilde{\omega}_t(i)^{\gamma_t}}{\sum_j \widetilde{\omega}_t(i)^{\gamma_t}}$

NTM version 2

DIFFERENTIABLE NEURAL COMPUTER

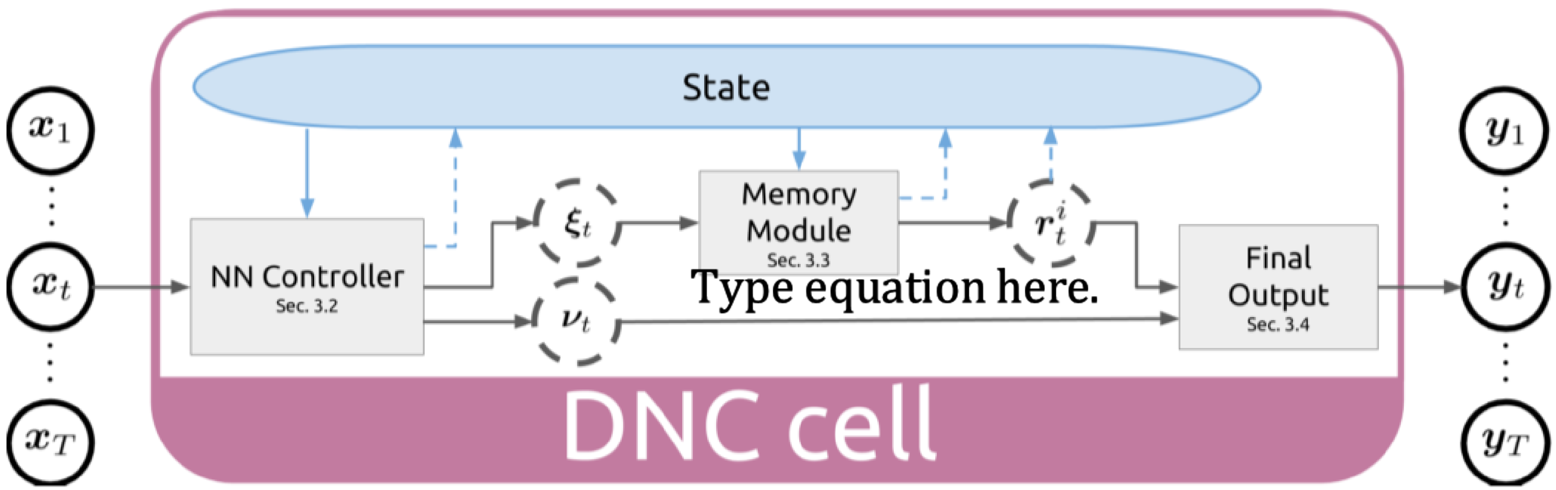
DNC



三种differentiable attention：

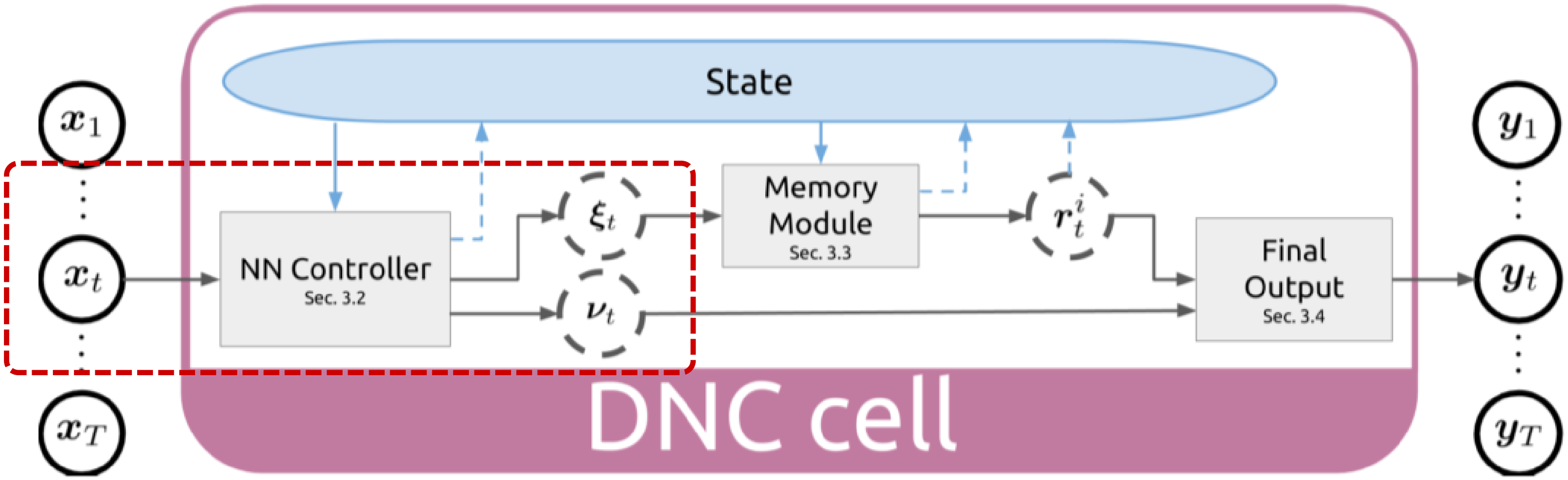
content addressing; temporal shift; memory usage

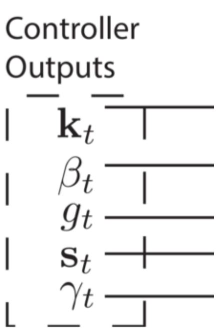
DNC



$$(output, new\_state) = DNC(input, state)$$

DNC: controller

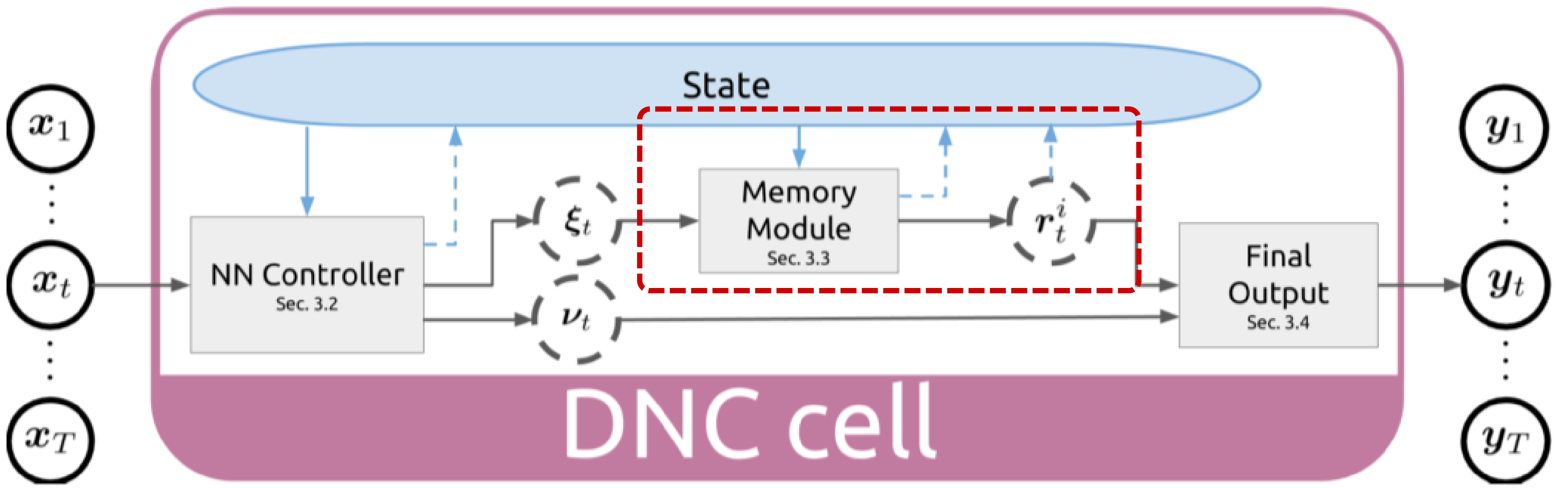




Controller产生interface parameters 和output parameters

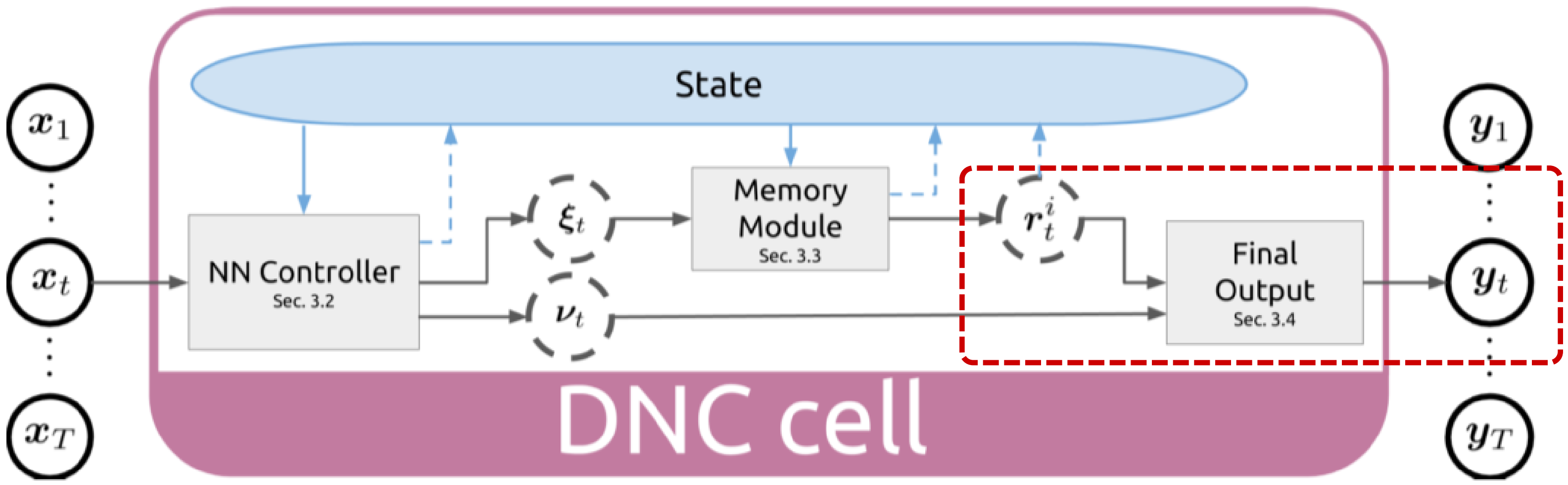
- $\chi_t = [x_t; r_{t-1}^1; \dots; r_{t-1}^R]$
- $(\xi_t, v_t) = NNC([\chi_1; \dots; \chi_t]; \theta)$

DNC: memory



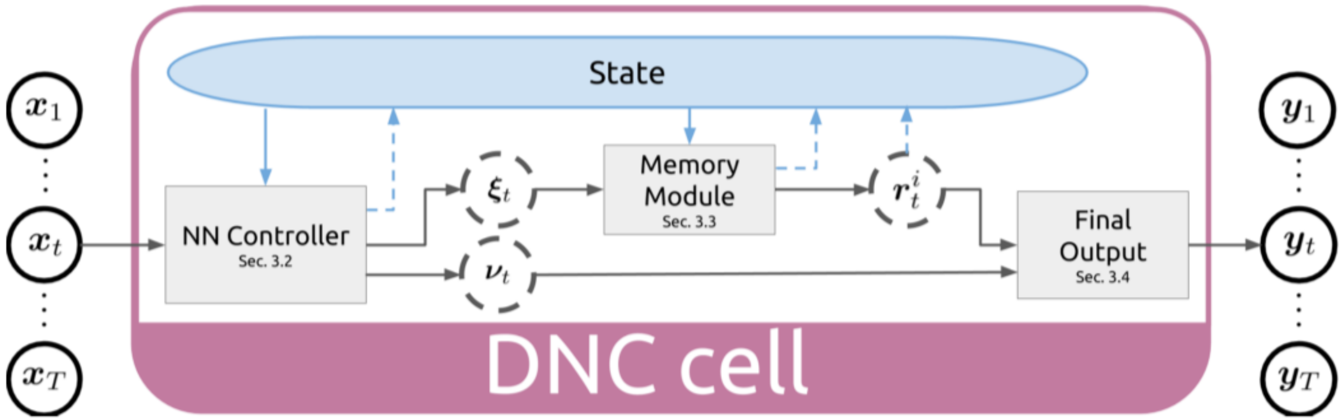
- Content based writing and reading weight
- History based writing weight => final writing weights
- History based reading weight => final reading weights

DNC: output

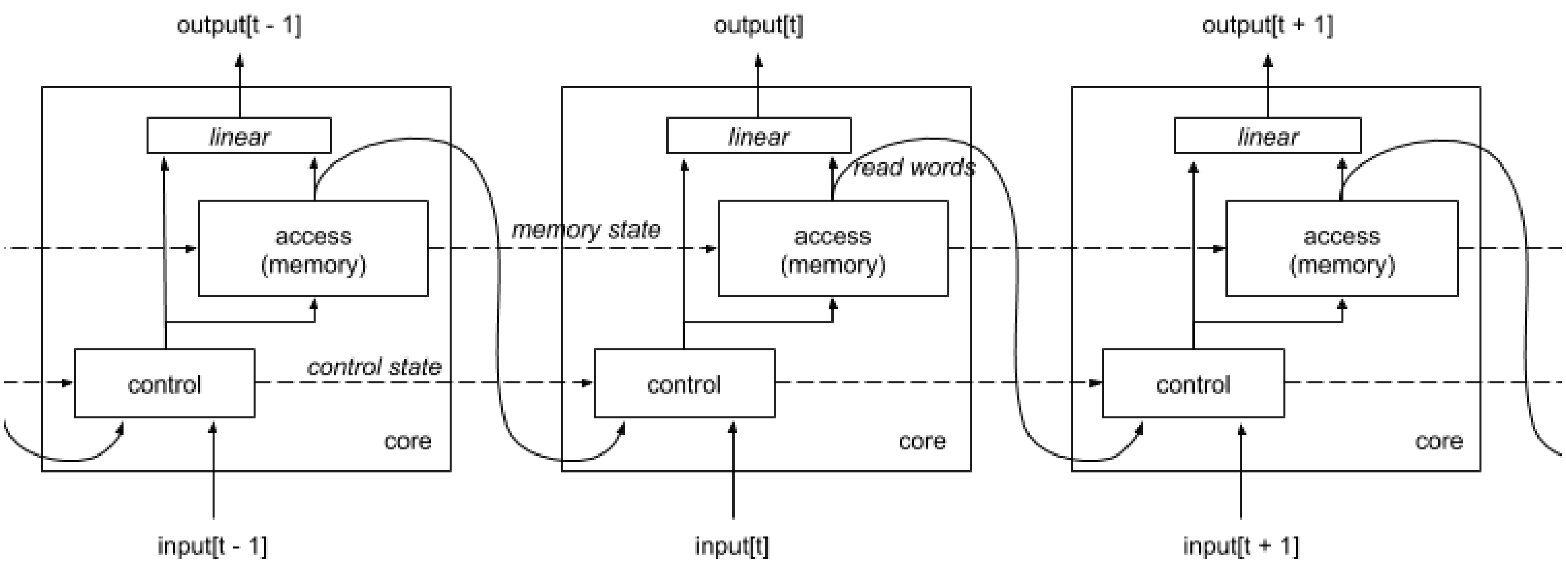


$$y_t = W_r\left[r_t^1, \dots r_t^R\right] + v_t$$

DNC: output

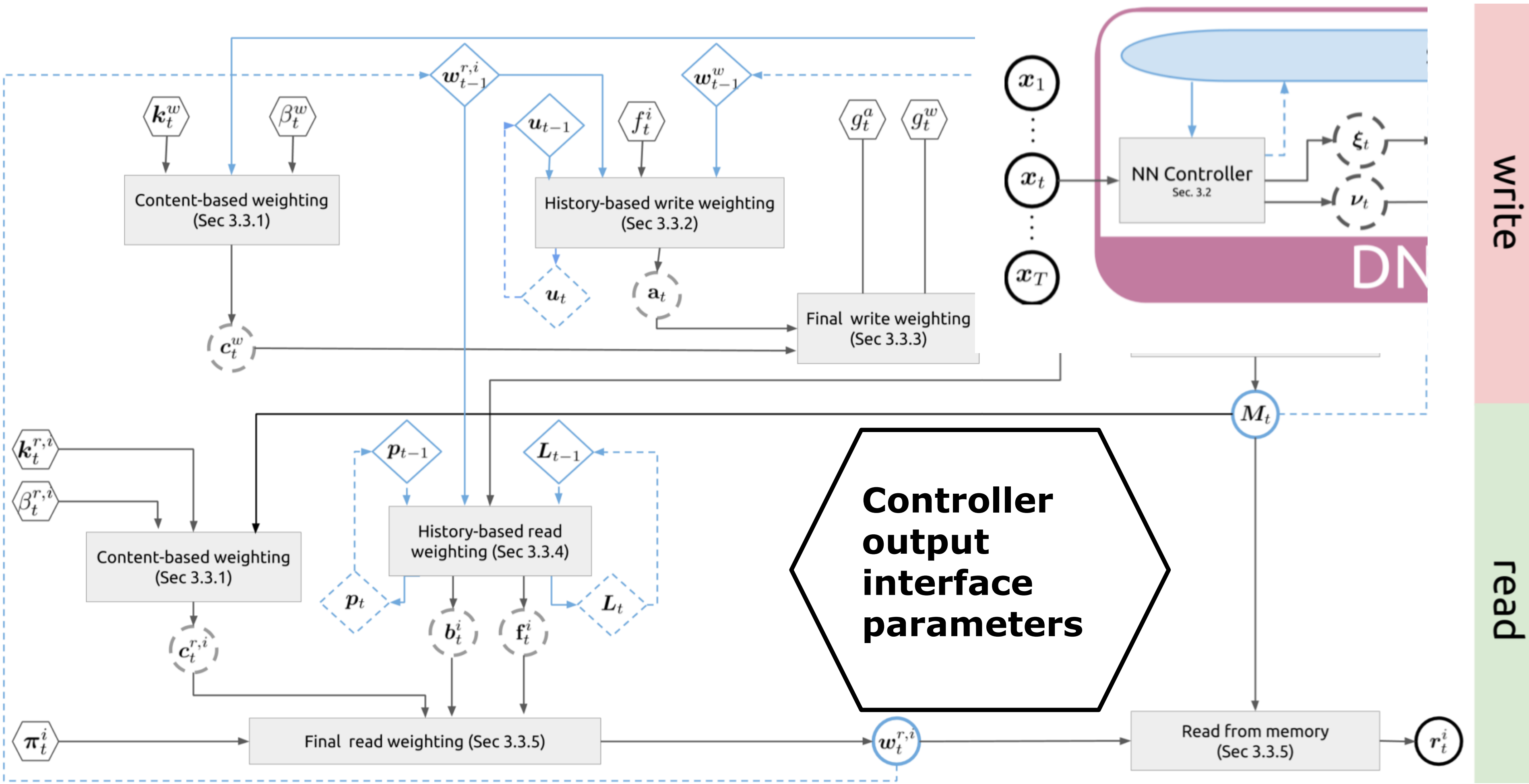


State
x_1
x_t
x_T
NN Controller
Sec. 3.2
ξ_t
ν_t
Memory Module
Sec. 3.3
r_t^i
Final Output
Sec. 3.4
y_1
y_t
y_T
DNC cell



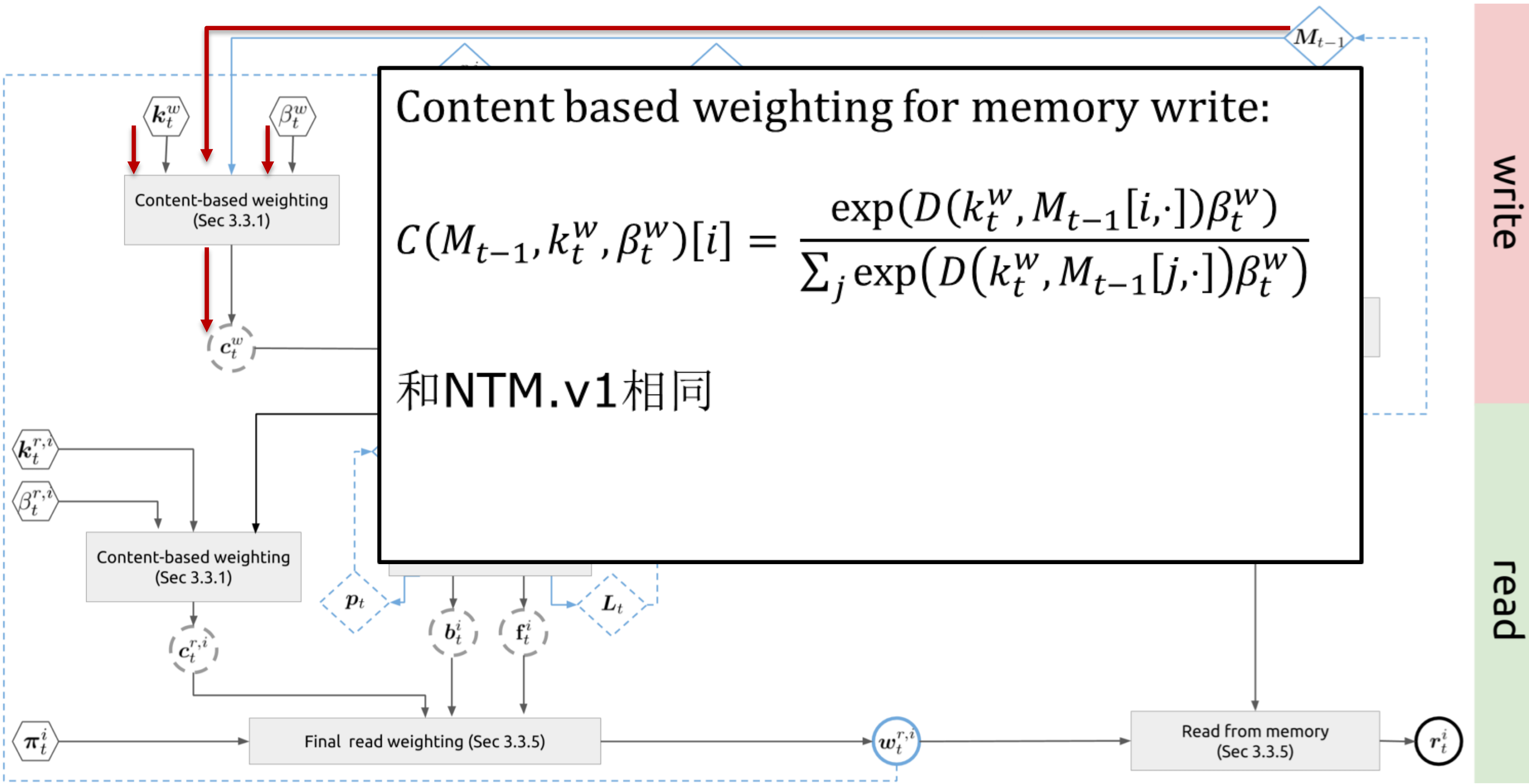
output[t - 1]
output[t]
output[t + 1]
linear
read words
access (memory)
memory state
control
control state
core
input[t - 1]
input[t]
input[t + 1]

DNC: details



Content-based weighting (Sec 3.3.1)
History-based write weighting (Sec 3.3.2)
Final write weighting (Sec 3.3.3)
NN Controller
Sec. 3.2
DN
write
read
Content-based weighting (Sec 3.3.1)
History-based read weighting (Sec 3.3.4)
Controller output interface parameters
Final read weighting (Sec 3.3.5)
Read from memory (Sec 3.3.5)

DNC: details

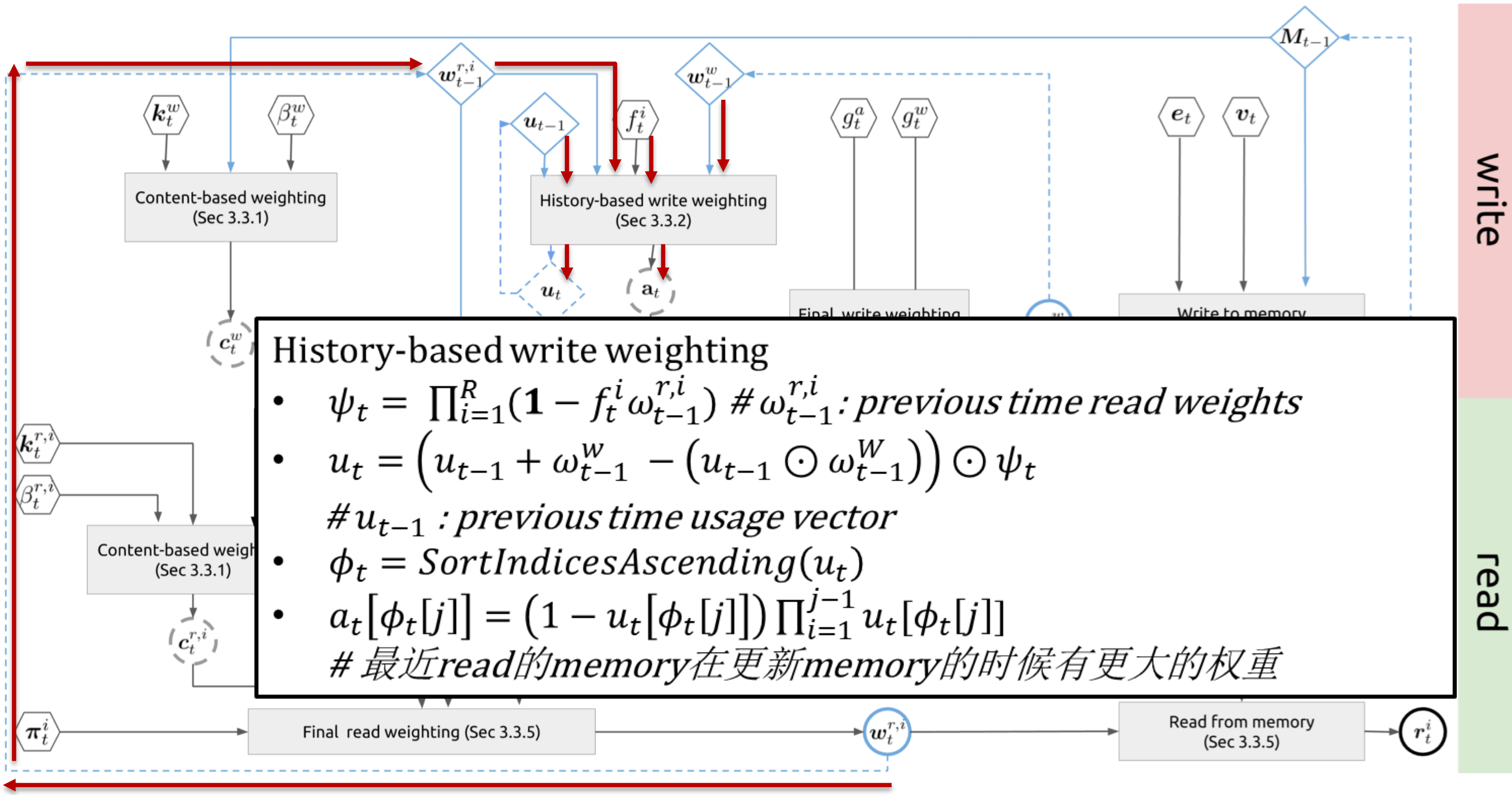


Content based weighting for memory write:

$$C(M_{t-1}, k_t^w, \beta_t^w)[i] = \frac{\exp(D(k_t^w, M_{t-1}[i,\cdot])\beta_t^w)}{\sum_j \exp(D(k_t^w, M_{t-1}[j,\cdot])\beta_t^w)}$$

和NTM.v1相同

DNC: details



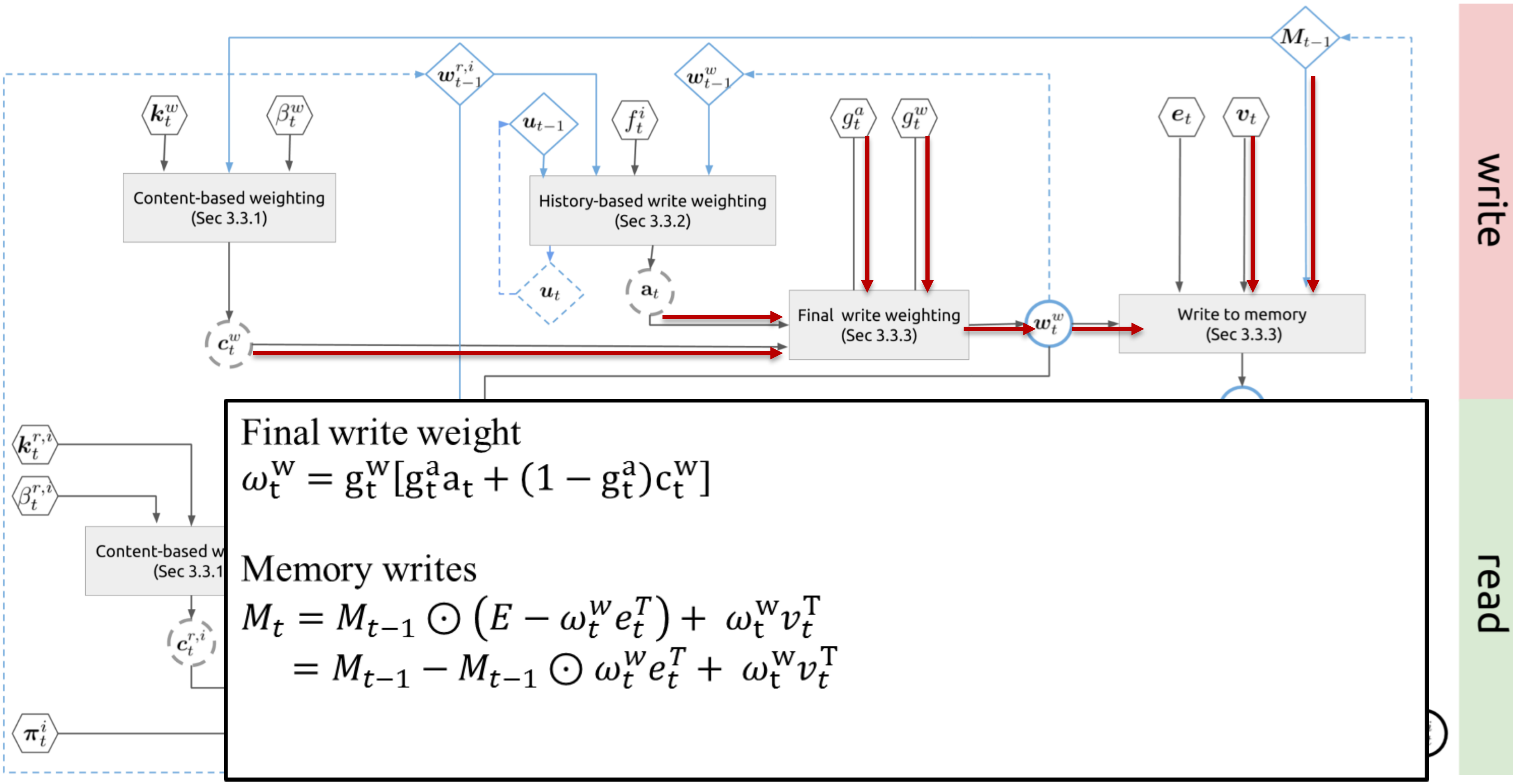
History-based write weighting

- $\psi_t = \prod_{i=1}^{R}(\mathbf{1} - f_t^i \omega_{t-1}^{r,i})$ $\# \omega_{t-1}^{r,i}: previous\ time\ read\ weights$
- $u_t = \left(u_{t-1} + \omega_{t-1}^{W} - (u_{t-1} \odot \omega_{t-1}^{W})\right) \odot \psi_t$

  $\# u_{t-1}: previous\ time\ usage\ vector$
- $\phi_t = SortIndicesAscending(u_t)$
- $a_t[\phi_t[j]] = (1 - u_t[\phi_t[j]]) \prod_{i=1}^{j-1} u_t[\phi_t[j]]$

  # 最近read的memory在更新memory的时候有更大的权重

DNC: details



Final write weight

$$\omega_t^{w} = g_t^{w}[g_t^{a} a_t + (1 - g_t^{a}) c_t^{w}]$$

Memory writes

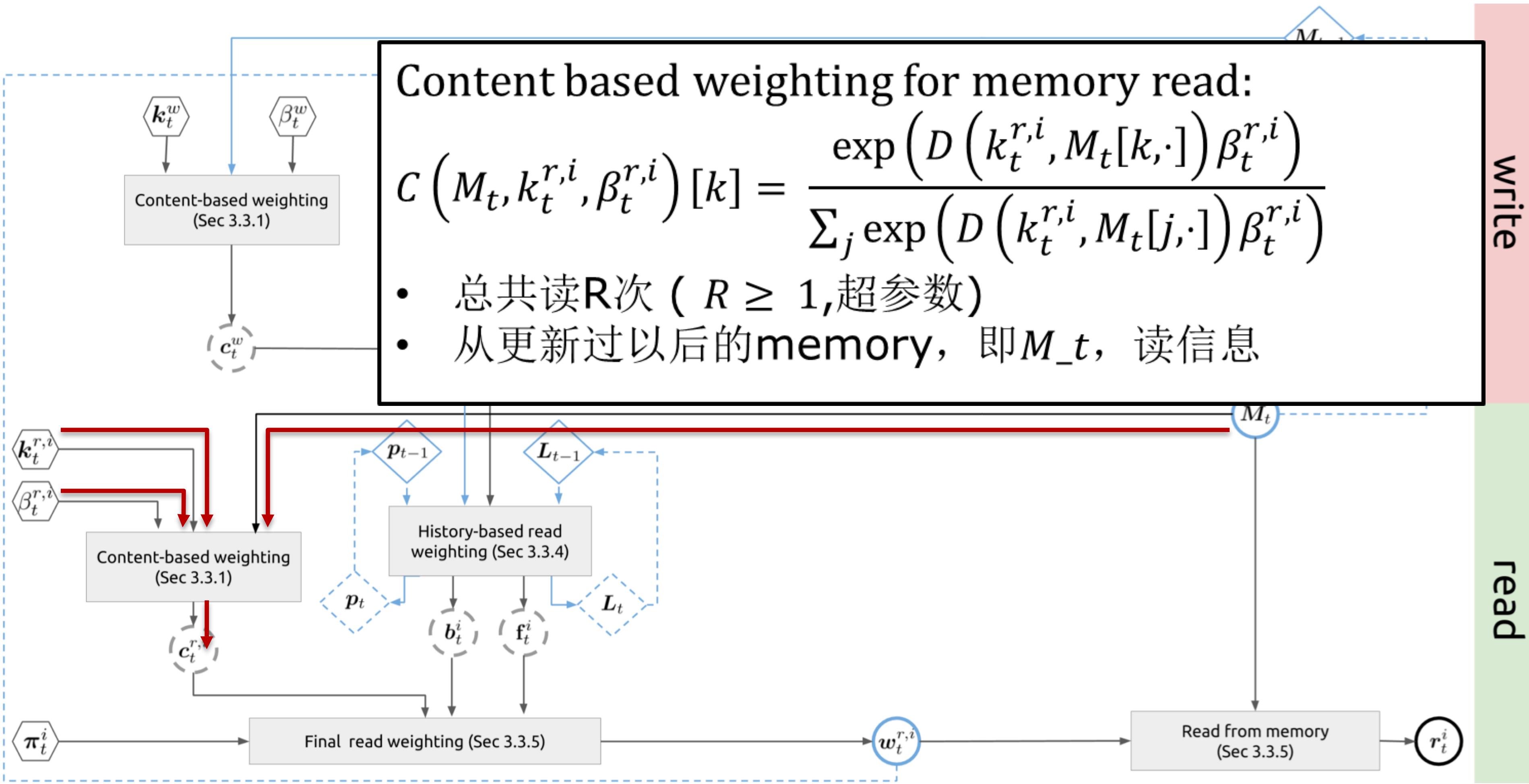
$$M_t = M_{t-1} \odot \left(E - \omega_t^{w} e_t^{T}\right) + \omega_t^{w} v_t^{T}$$
$$= M_{t-1} - M_{t-1} \odot \omega_t^{w} e_t^{T} + \omega_t^{w} v_t^{T}$$

DNC: details

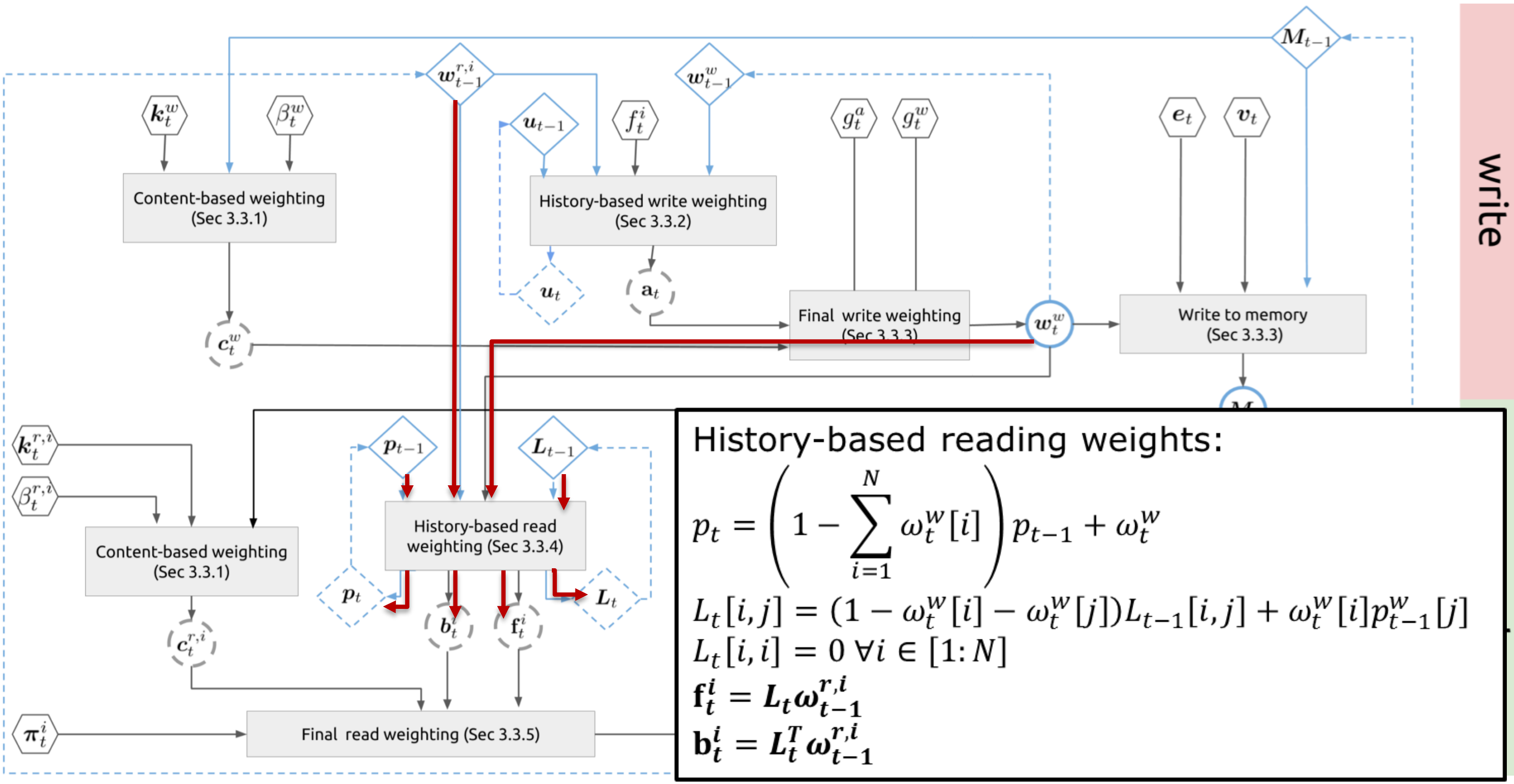
Content based weighting for memory read:

$$C\left(M_t, k_t^{r,i}, \beta_t^{r,i}\right)[k] = \frac{\exp\left(D\left(k_t^{r,i}, M_t[k,\cdot]\right)\beta_t^{r,i}\right)}{\sum_j \exp\left(D\left(k_t^{r,i}, M_t[j,\cdot]\right)\beta_t^{r,i}\right)}$$

- 总共读R次（ $R \geq 1$,超参数)
- 从更新过以后的memory，即M_t，读信息



DNC: details



History-based reading weights:

$$p_t = \left(1 - \sum_{i=1}^{N} \omega_t^w[i]\right) p_{t-1} + \omega_t^w$$

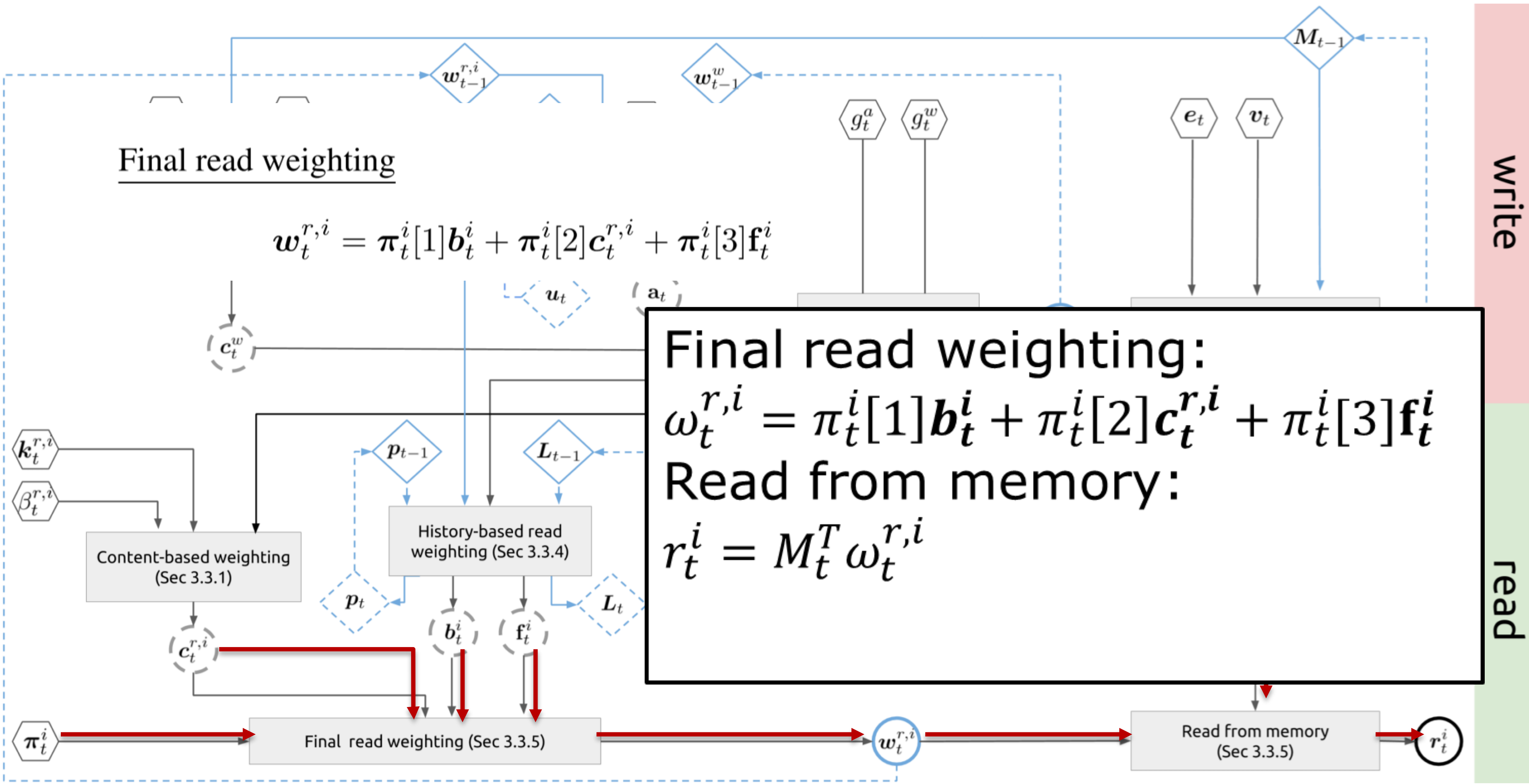
$$L_t[i,j] = (1 - \omega_t^w[i] - \omega_t^w[j]) L_{t-1}[i,j] + \omega_t^w[i] p_{t-1}^w[j]$$

$$L_t[i,i] = 0 \ \forall i \in [1:N]$$

$$\mathbf{f}_t^i = \boldsymbol{L}_t \boldsymbol{\omega}_{t-1}^{r,i}$$

$$\mathbf{b}_t^i = \boldsymbol{L}_t^T \boldsymbol{\omega}_{t-1}^{r,i}$$

DNC: details



Final read weighting:

$$\omega_t^{r,i} = \pi_t^i[1]\boldsymbol{b}_t^i + \pi_t^i[2]\boldsymbol{c}_t^{r,i} + \pi_t^i[3]\mathbf{f}_t^i$$

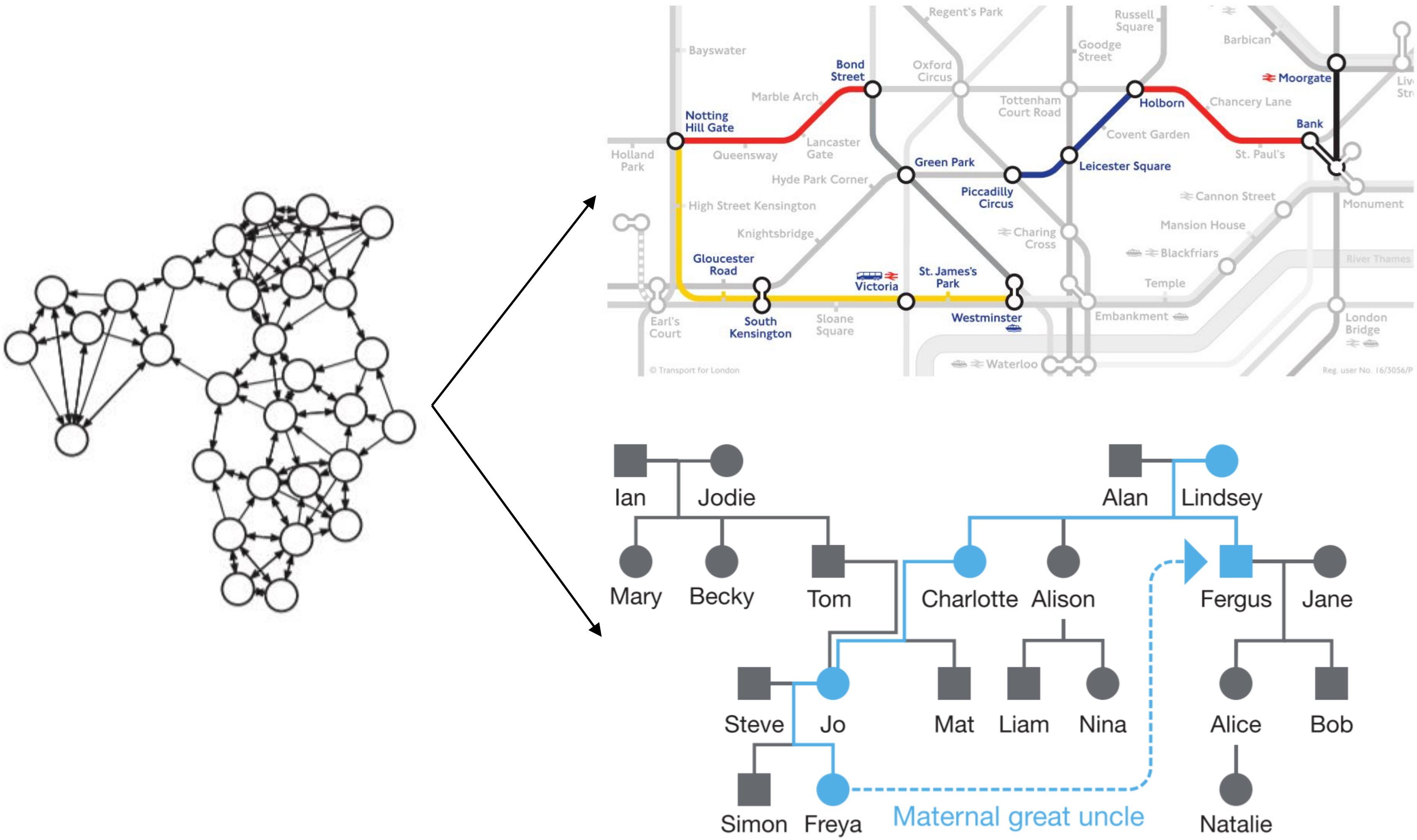
Read from memory:

$$r_t^i = M_t^T \omega_t^{r,i}$$

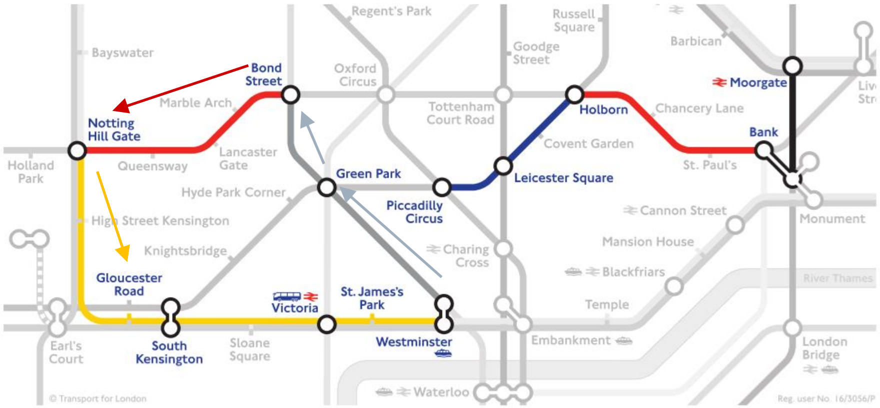
DNC 代码演示

- DNC + bAbI task

DNC 应用: graph experiment



DNC 应用: graph experiment



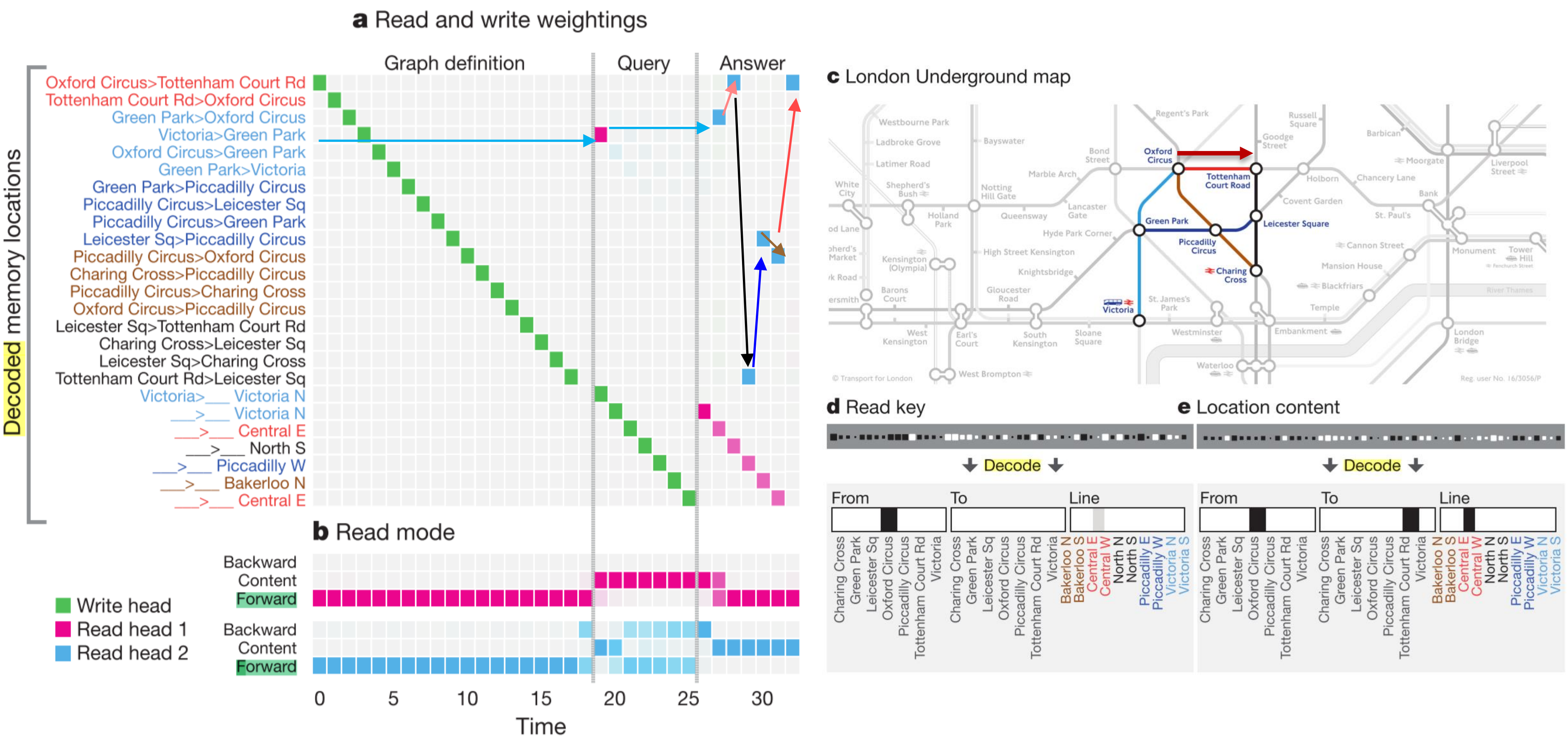
Underground input:
(OxfordCircus, TottenhamCtRd, Central)
(TottenhamCtRd, OxfordCircus, Central)
(BakerSt, Marylebone, Circle)
(BakerSt, Marylebone, Bakerloo)
(BakerSt, OxfordCircus, Bakerloo)
⋮
(LeicesterSq, CharingCross, Northern)
(TottenhamCtRd, LeicesterSq, Northern)
(OxfordCircus, PiccadillyCircus, Bakerloo)
(OxfordCircus, NottingHillGate, Central)
(OxfordCircus, Euston, Victoria)

84 edges in total

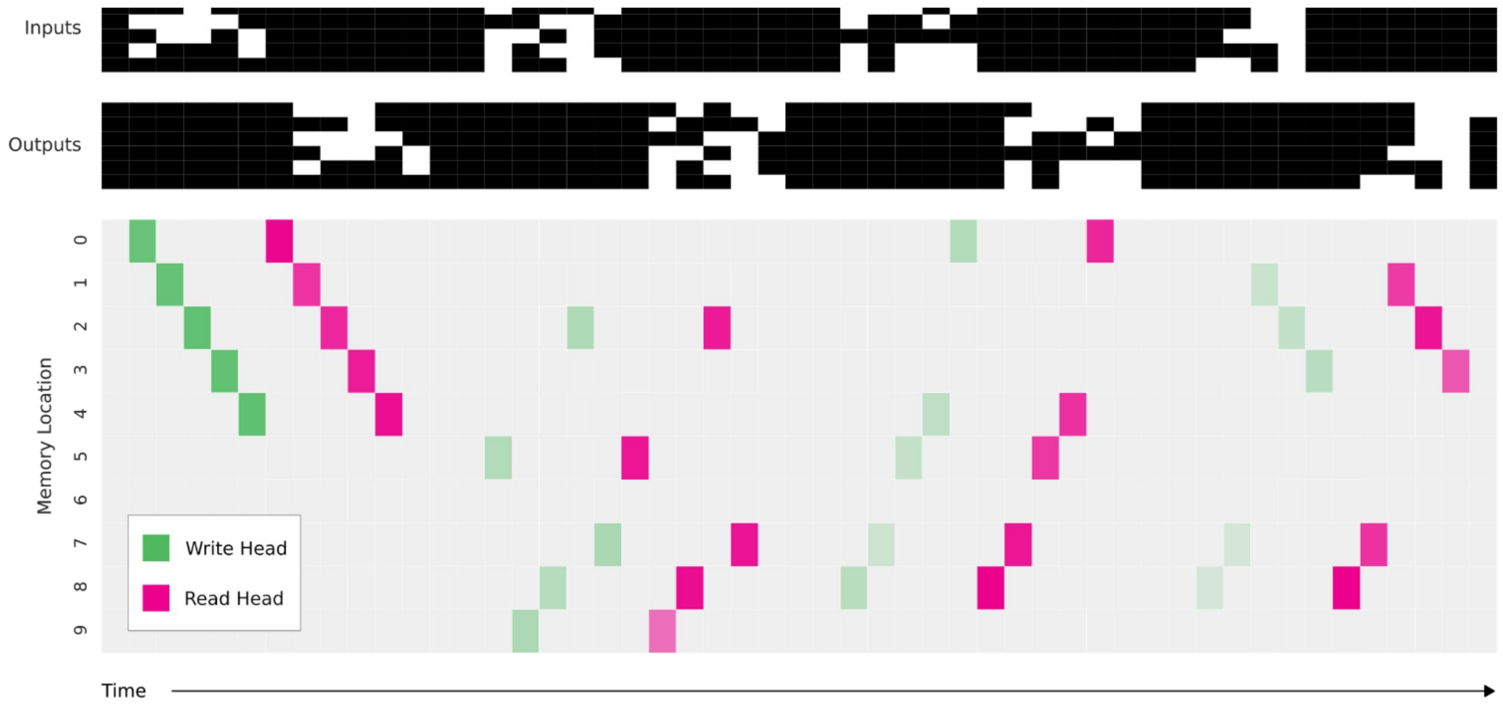
Traversal question:
(BondSt, _, Central),
(_, _, Circle), (_, _, Circle),
(_, _, Circle), (_, _, Circle),
(_, _, Jubilee), (_, _, Jubilee),

Answer:
(BondSt, NottingHillGate, Central)
(NottingHillGate, GloucesterRd, Circle)
⋮
(Westminster, GreenPark, Jubilee)
(GreenPark, BondSt, Jubilee)

DNC 应用: graph experiment

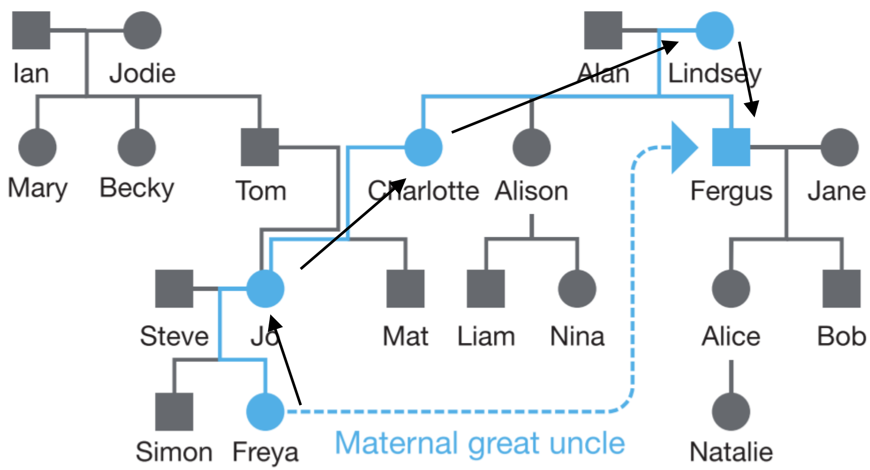


DNC 应用: graph experiment



使用copy task 帮助理解上一页的graph task

DNC 应用: graph experiment



Family tree input:

(Charlotte, Alan, Father)
(Simon, Steve, Father)
(Steve , Simon, Son1)
(Nina, Alison, Mother)
(Lindsey, Fergus, Son1)
⋮
(Bob, Jane, Mother)
(Natalie, Alice, Mother)
(Mary, Ian, Father)
(Jane, Alice, Daughter1)
(Mat, Charlotte, Mother)

54 edges in total

Inference question:

(Freya, _, MaternalGreatUncle)

Answer:

(Freya, Fergus, MaternalGreatUncle)

REASONING

Relational reasoning

- 目前的Deep Learning模型在feature learning方面有很好的表现，但是尚且不擅长推理
  - e.g. memory network*S* on bAbI task 17, 19
- 这不是Deep Learning不适合推理任务，而是尚不存在正确的深层学习架构或模块来实现一般的关系推理。例如，卷积神经网络在理解本地空间结构的能力上是无与伦比的 - 这就是为什么它们在图像识别模型中常用的 - 但是可能在其他推理任务中挣扎 (来源)

Relational Network

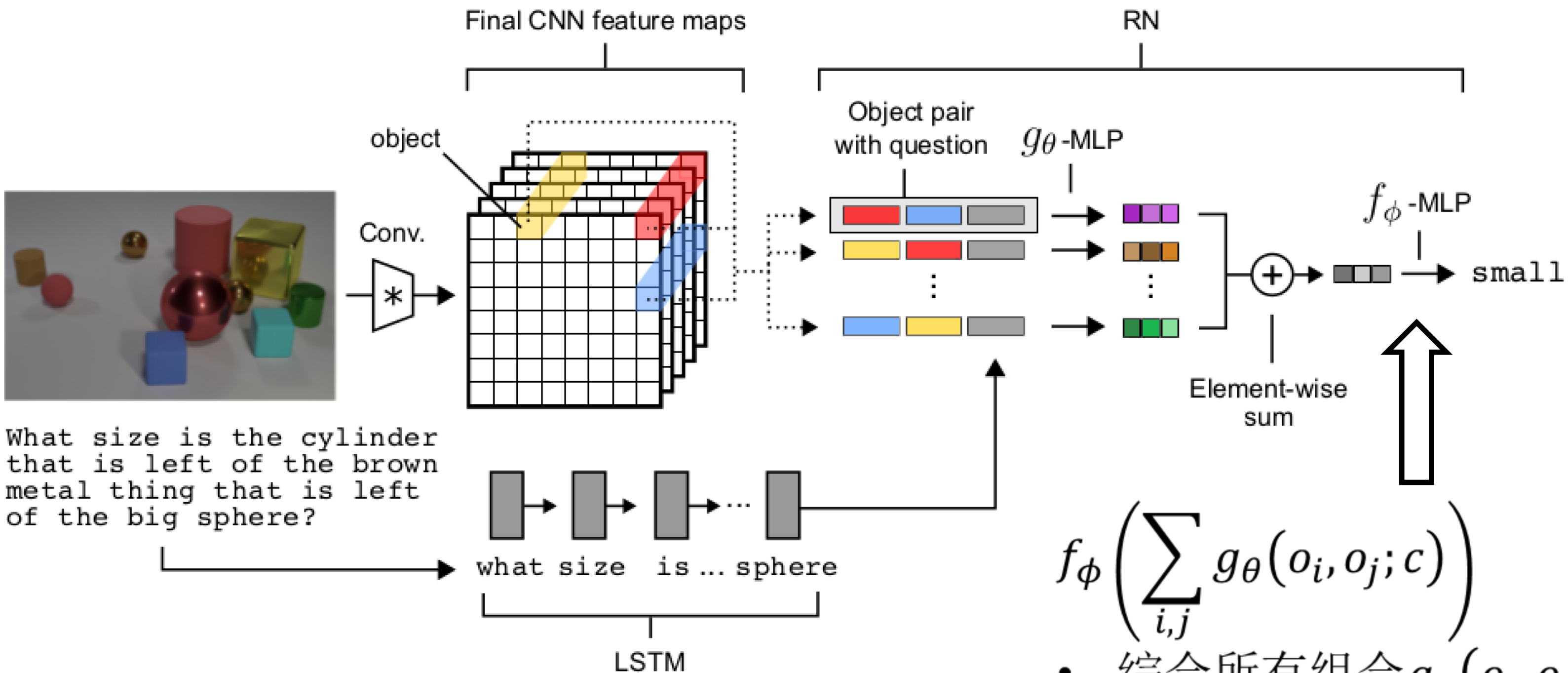
- A simple neural network module for relational reasoning (2017)
- Relational network (RN) 是一个用于relational reasoning的NN module
- $RN(O) = f_\phi\left(\sum_{i,j} g_\theta\left(o_i, o_j\right)\right)$
  - Inputs: $O = \{o_1, \dots o_n\}$
  - MLPs: $\mathrm{f}_\phi, g_\theta$
    - $g_\theta\left(o_i, o_j\right)$：使用NN量化$\mathrm{o}_i$ 和o_j的relation
    - $f_\phi\left(\sum_{i,j} g_\theta\left(o_i, o_j\right)\right)$：RN需要考虑所有组合的关系

Learn to infer relations

- $g_\theta(o_i, o_j)$：使用NN量化o_i 和o_j的relation
  - 任意两个对象之间的关系使用同一套参数$g_\theta(\cdot,\cdot)$
- $f_\phi(\sum_{i,j} g_\theta(o_i, o_j))$：
  - RN需要综合考虑所有组合的关系做出预测
  - RN *learn to infer* the existence and implications of object relations

Relational Network应用

"plug-and-play" modules



$$f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j; c)\right)$$

- Word-embedding-dim32; LSTM-dim128
- g_θ: 4-layer MLP, dim256 per layer, RELU
- f_ϕ: 3-layer MLP, dim-256-256-29, RELU

- 综合所有组合$g_\theta(o_i, o_j; c)$
- *implicitly*提取有用的组合
- 预测最终答案

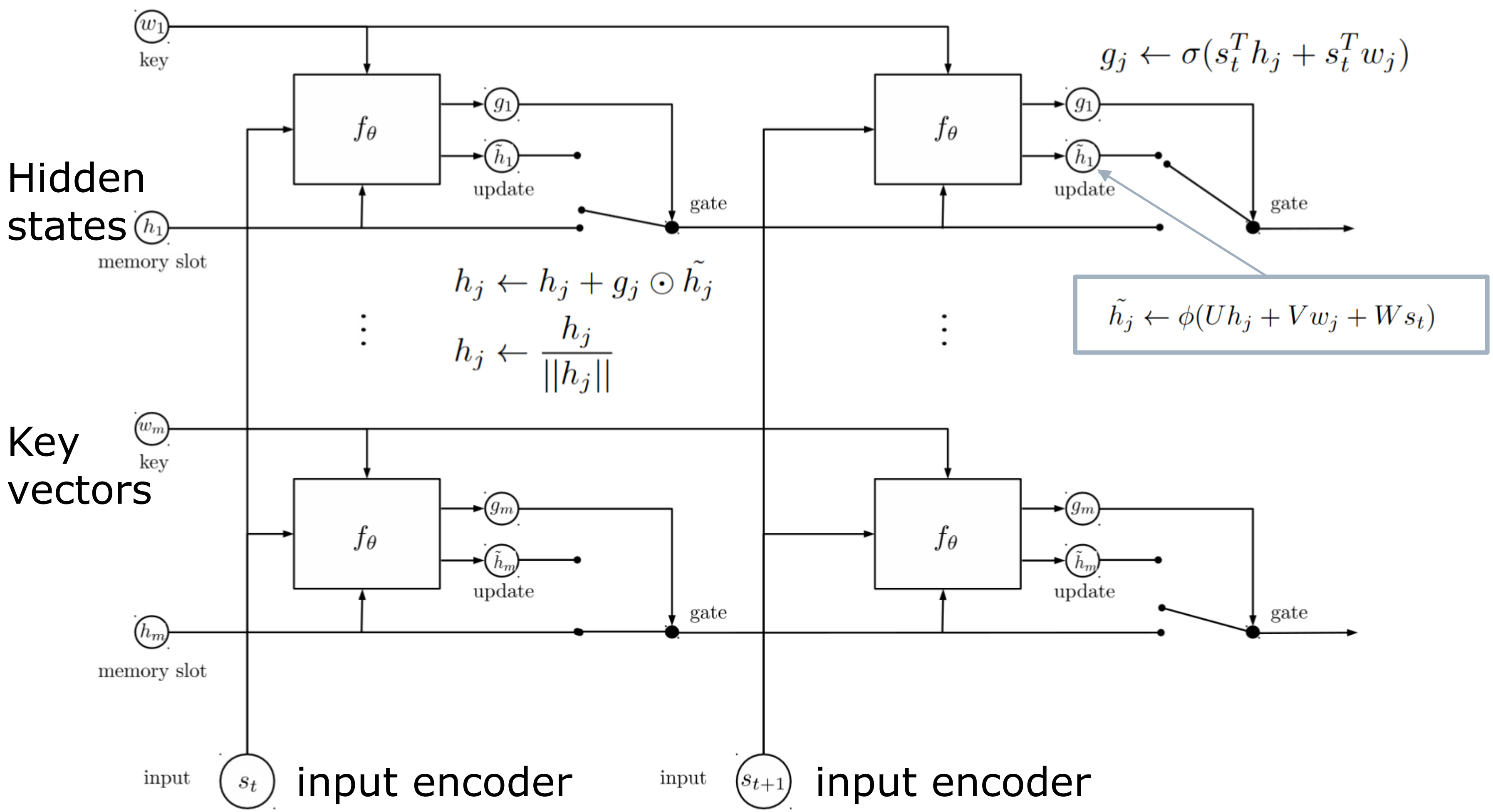
Relational Network应用

- 实验表明RN module在以bAbI为例的文本QA任务上有和Memory network相似的表现
- bAbI experiment:
  - 每个问题之前的最多20句话挑选做support set
  - 使用dim-32-LSTM的encoding state把support set转化为RN的object set
  - 使用另一个dim-32-LSTM的encoding state表示问题
  - g_θ使用256×4的四层MLP
  - f_ϕ使用$256, 512, 159$的三层MLP
- Joint training 情况下，通过18/20 bAbI test
  - 在task16，basic induction可以达到2.1%的误差水平
  - 由于DNC55.1%，EntNet52.1%

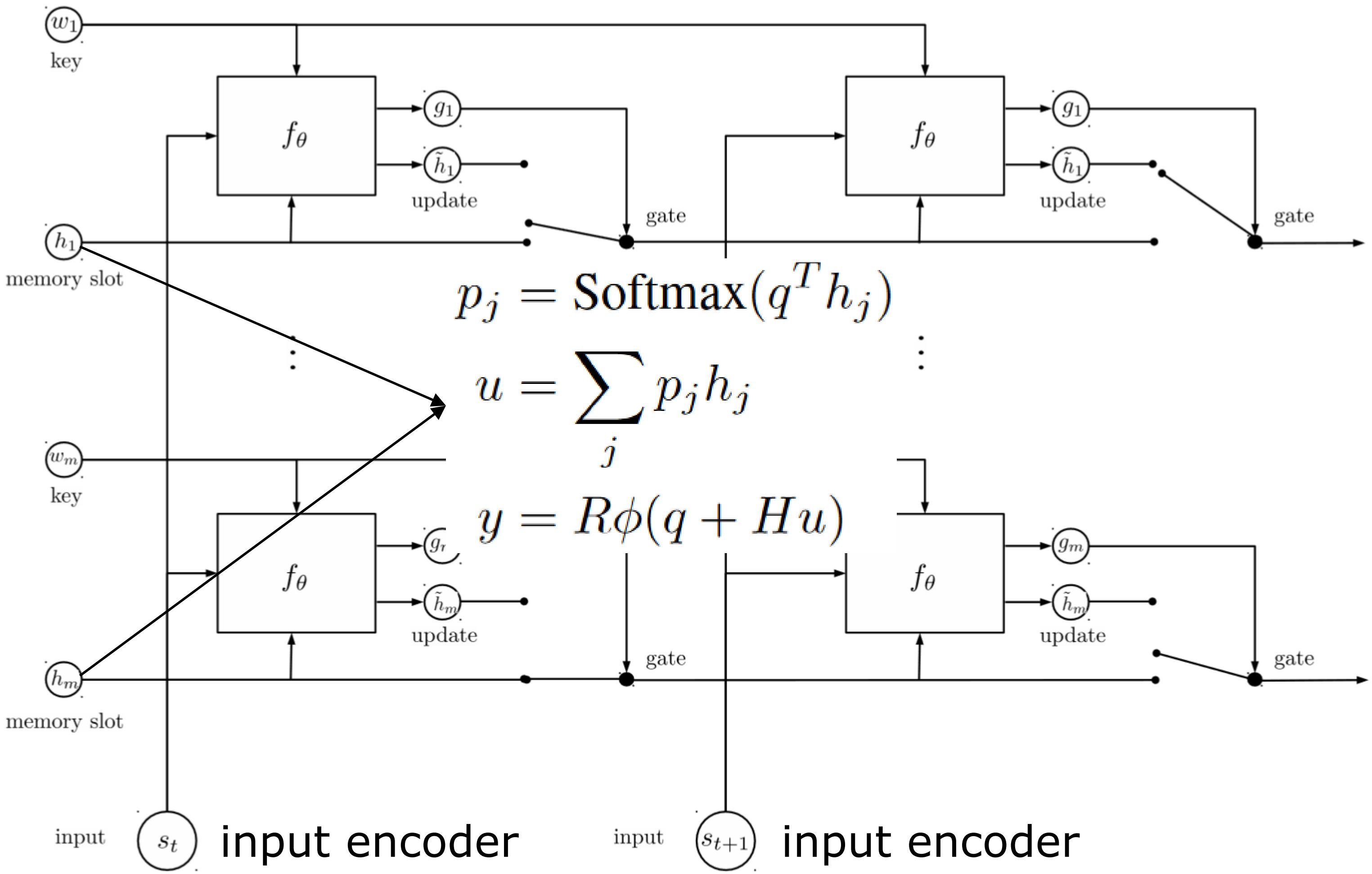
Recurrent Entity Network

- Tracking the world state with Recurrent Entity Network (2017)
- REN 使用一个dynamic long-term memory 记忆和更新信
  - 固定大小的记忆
  - 使用location-based 和 content-based 方式读&写记忆
  - 由一个 input encoder, 一个dynamic memory 和一个 output layer构成

Recurrent Entity Network: dynamic memory



Recurrent Entity Network: output module

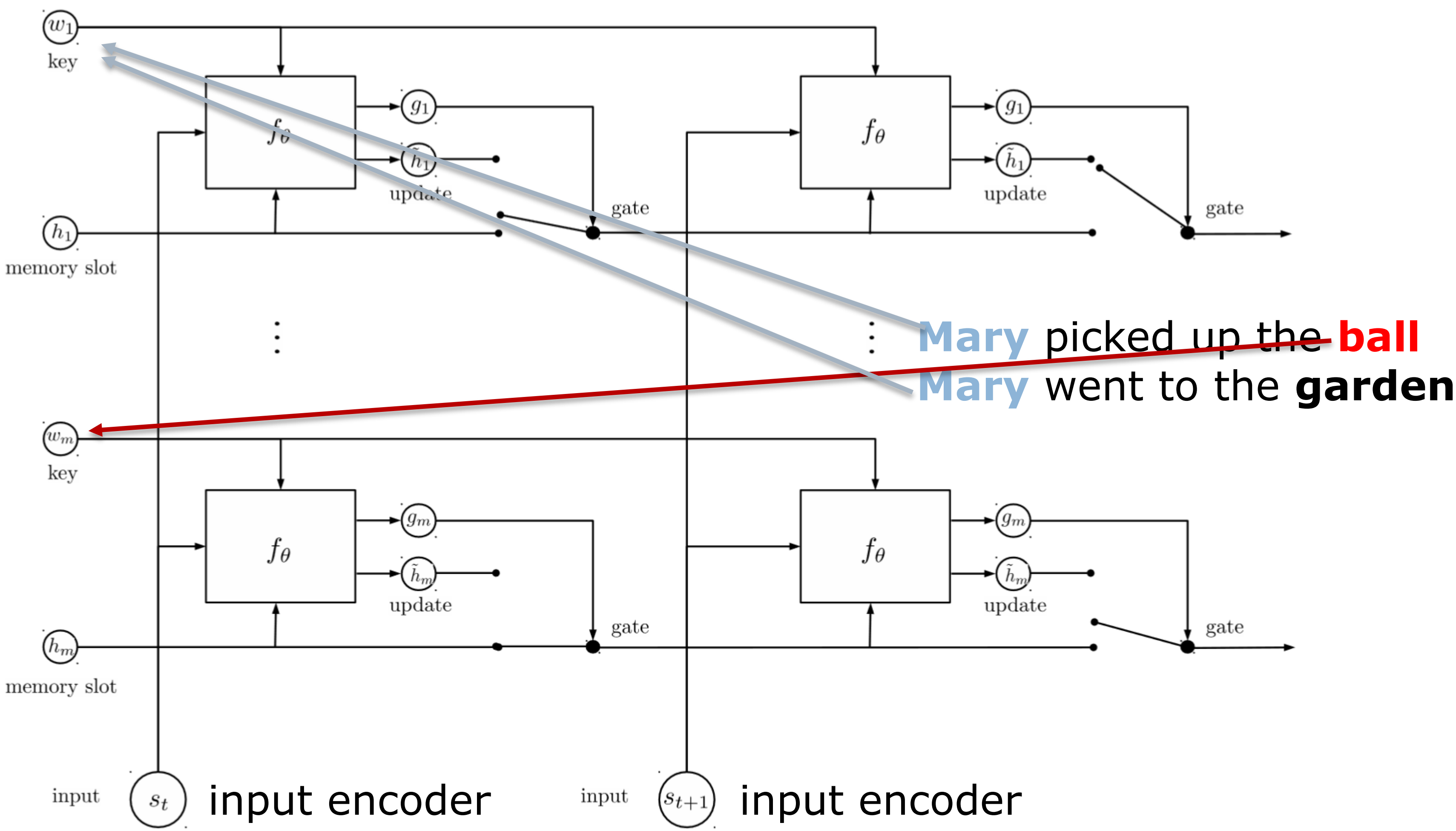


$$p_j = \mathrm{Softmax}(q^T h_j)$$

$$u = \sum_j p_j h_j$$

$$y = R\phi(q + Hu)$$

Recurrent Entity Network: intuition



Recurrent Entity Network: performance

- bAbI en-10K:
  - 如果对20个测试分别训练，可以通过所有的测试
  - 如果对20个测试同时训练（joint training)，可以通过16个测试（RN 可以通过18个测试）
    - 3 supporting facts； basic induction；positional reasoning；path finding

Recurrent Entity Network: performance

- bAbI en-10K:
  - 初始化问题：参数随机初始化10次，训练10个模型，选择表现最好的那个
    - 相对于DNC，不同初始化参数的variance更小

Table 7: Results on bAbI Tasks with 10k samples and joint training on all tasks.

	All Seeds		Best Seed	
Task	DNC	EntNet	DNC	EntNet
1: 1 supporting fact	9.0 ± 12.6	0 ± 0.1	0	0.1
2: 2 supporting facts	39.2 ± 20.5	15.3 ± 15.7	0.4	2.8
3: 3 supporting facts	39.6 ± 16.4	29.3 ± 26.3	1.8	10.6
19: path finding	64.6 ± 37.4	70.4 ± 6.1	3.9	63.0
20: agent's motivation	0.0 ± 0.1	0 ± 0	0	0
Failed Tasks (> 5%):	11.2 ± 5.4	$\mathbf{5 \pm 1.2}$	**2**	4
Mean Error:	16.7 ± 7.6	$\mathbf{9.7 \pm 2.6}$	**3.8**	7.38

疑问

- 问题答疑： http://www.xxwenda.com/
  - 可邀请老师或者其他人回答问题

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop