

《神经网络与深度学习》



循环神经网络

<https://nndl.github.io/>

参考资料

- ▶ 《神经网络与深度学习》 第 6 章

- ▶ <https://nndl.github.io/>

- ▶ 网络资料

- ▶ An Introduction to Recurrent Neural Networks

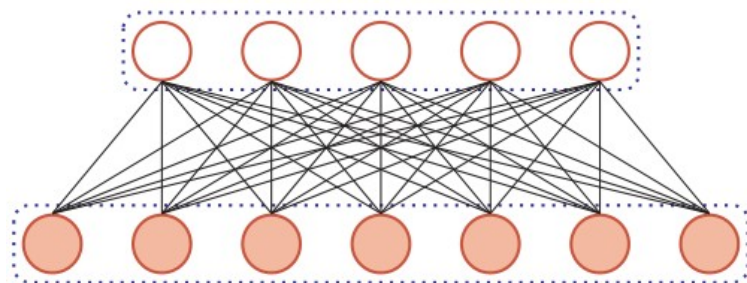
- ▶ <https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912>

- ▶ Recurrent Neural Networks

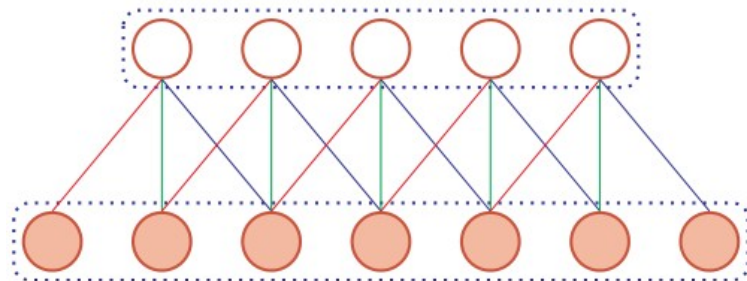
- ▶ <https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce>

前馈网络

- ▶ 连接存在层与层之间，每层的节点之间是无连接的。（无循环）
- ▶ 输入和输出的维数都是固定的，不能任意改变。无法处理变长的序列数据。



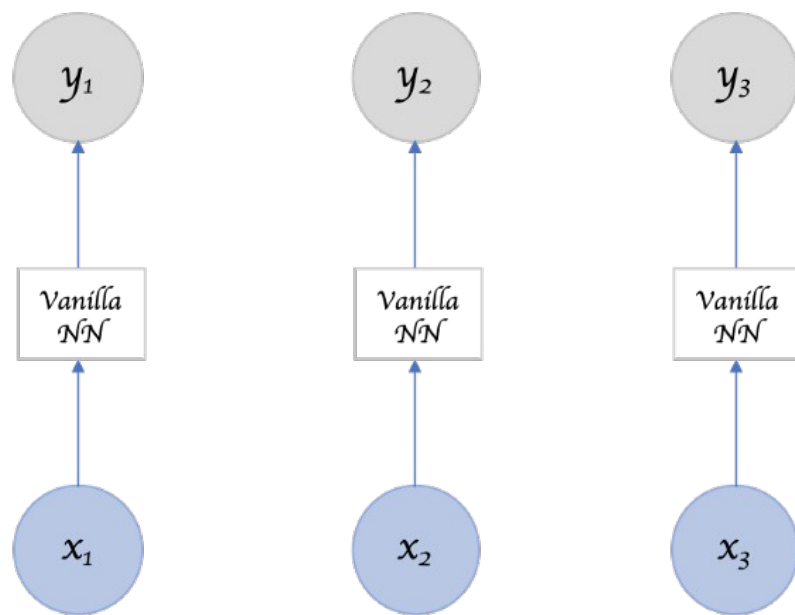
(a) 全连接层



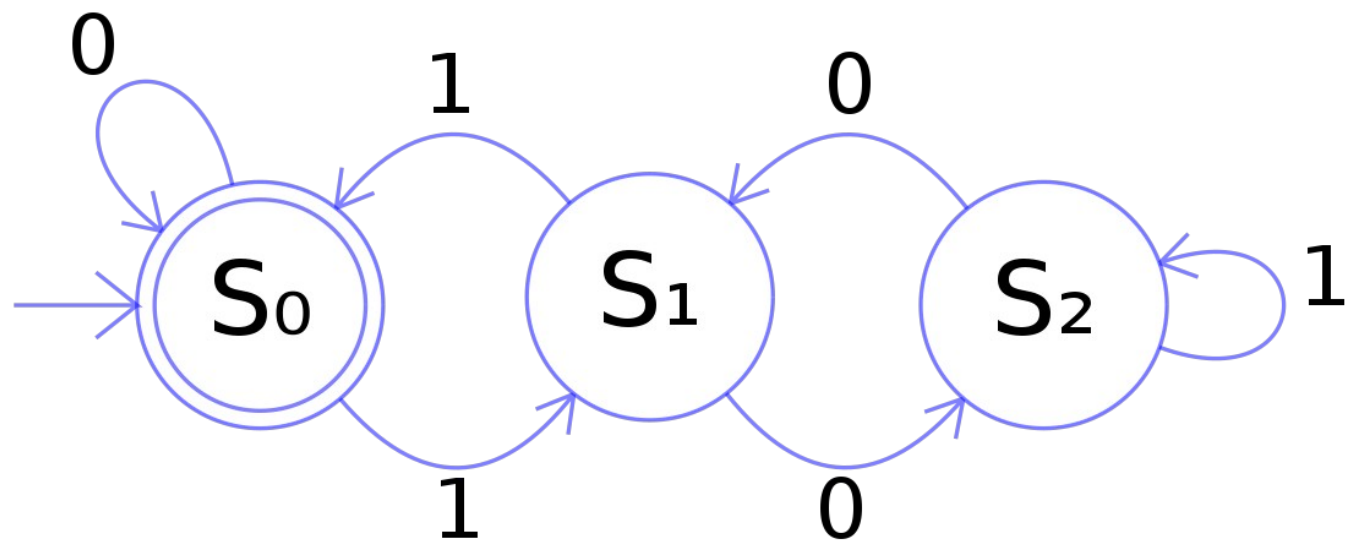
(b) 卷积层

前馈网络

- ▶ 假设每次输入都是独立的，也就是说每次网络的输出只依赖于当前的输入。



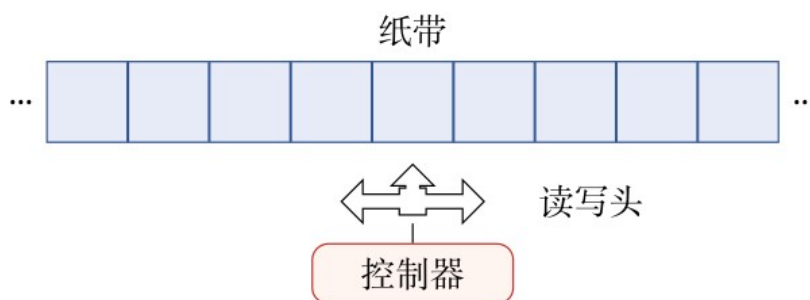
有限状态自动机 (Finite Automata)



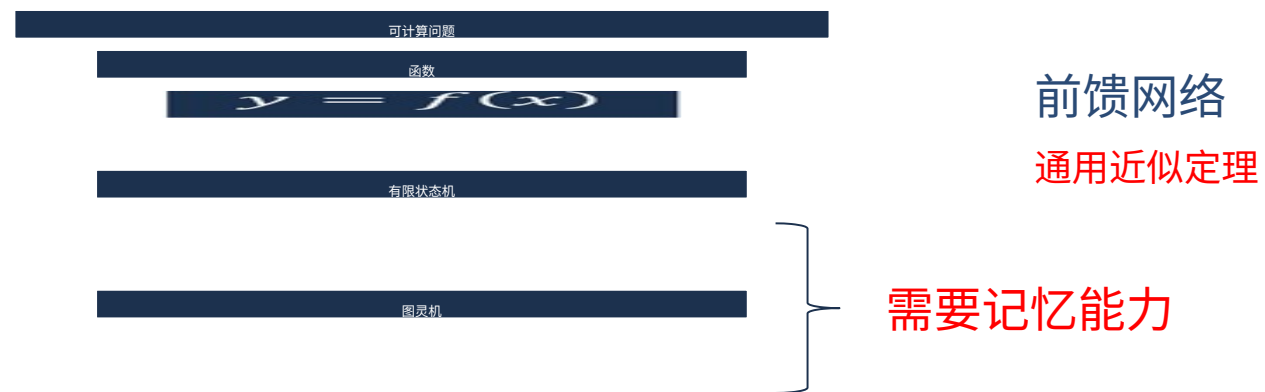
如何用 FNN 去模拟一个有限状态自动机？

图灵机

- ▶ 一种抽象数学模型，可以用来模拟任何可计算问题。



可计算问题



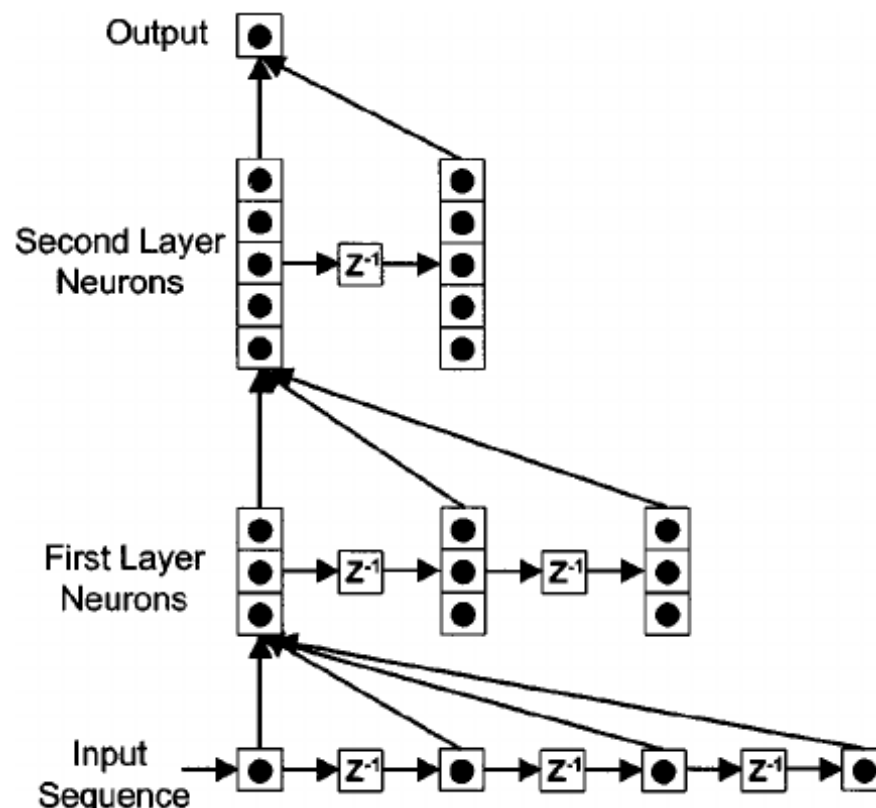
如何给网络增加记忆能力?

如何给网络增加记忆能力？

- ▶ **延时神经网络（Time Delay Neural Network，TDNN）**
- ▶ 建立一个额外的延时单元，用来存储网络的历史信息（可以包括输入、输出、隐状态等）

$$\mathbf{h}_t^{(l)} = f(\mathbf{h}_t^{(l-1)}, \mathbf{h}_{t-1}^{(l-1)}, \dots, \mathbf{h}_{t-K}^{(l-1)})$$

- ▶ 这样，前馈网络就具有了短期记忆的能力。



https://www.researchgate.net/publication/12314435_Neural_system_identification_model_of_human_sound_localization

如何给网络增加记忆能力?

▶ 自回归模型 (Autoregressive Model, AR)

- ▶ 一类时间序列模型，用亦是自己过去的信息来预测自己

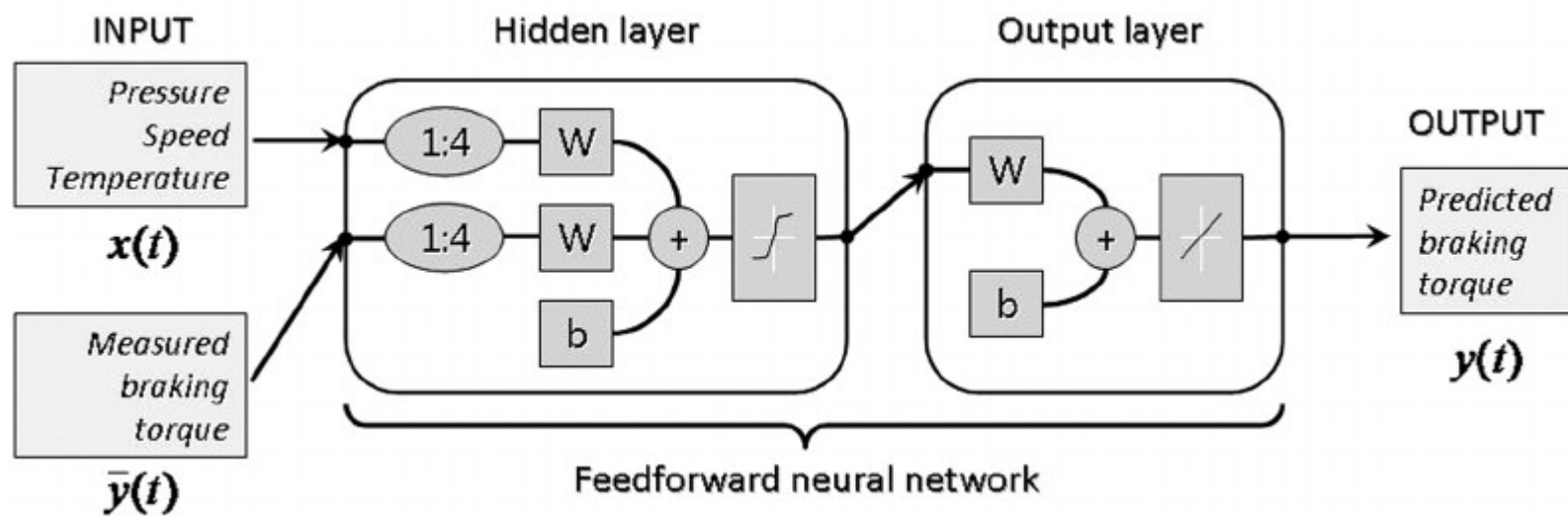
$$\mathbf{y}_t = w_0 + \sum_{k=1}^K w_k \mathbf{y}_{t-k} + \epsilon_t$$

- ▶ $\epsilon_t \sim N(0, \sigma^2)$ 为第 t 个时刻的噪声
- ▶ 有外部输入的非线性自回归模型 (Nonlinear Autoregressive with Exogenous Inputs Model, NARX)

$$\mathbf{y}_t = f(\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_{t-K_x}, \mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \dots, \mathbf{y}_{t-K_y})$$

- ▶ 其中 $f(\cdot)$ 表示非线性函数，可以是一个前馈网络， K_x 和 K_y 为超参数。

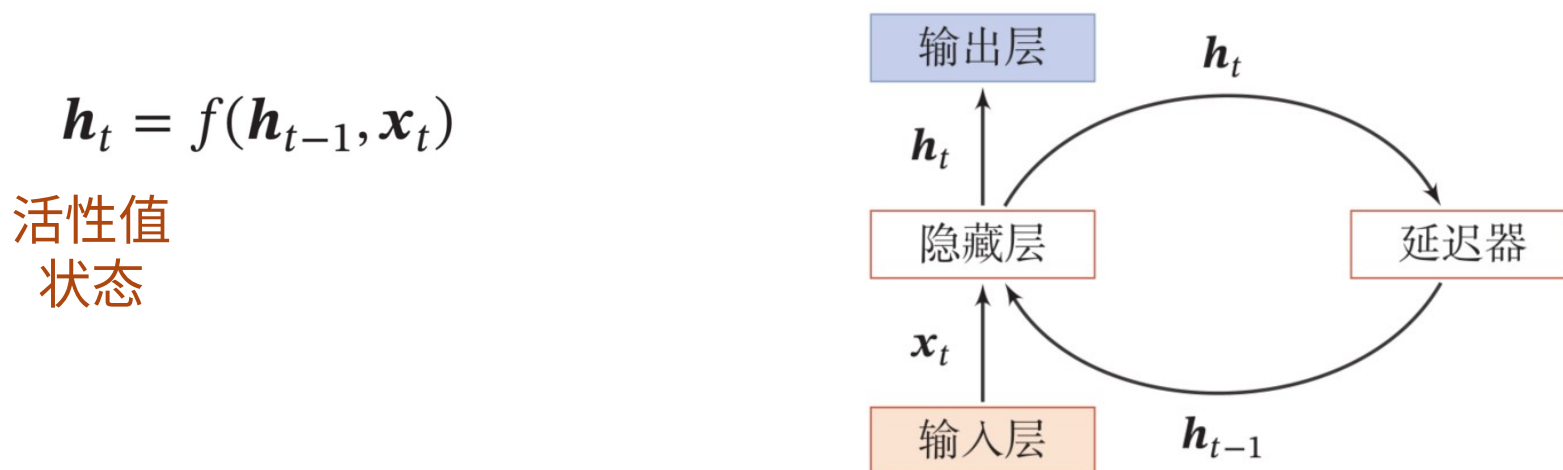
非线性自回归模型



https://www.researchgate.net/publication/234052442_Braking_torque_control_using_reccurent_neural_networks

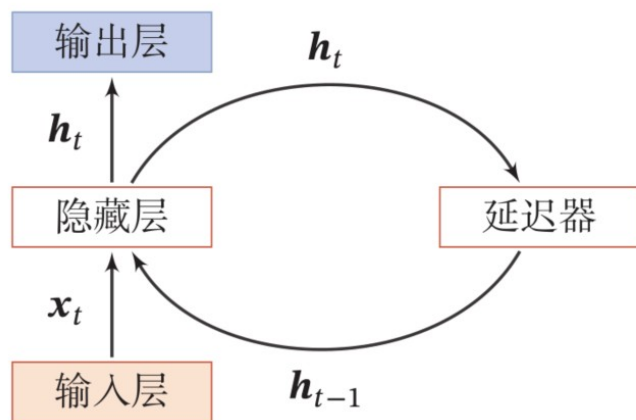
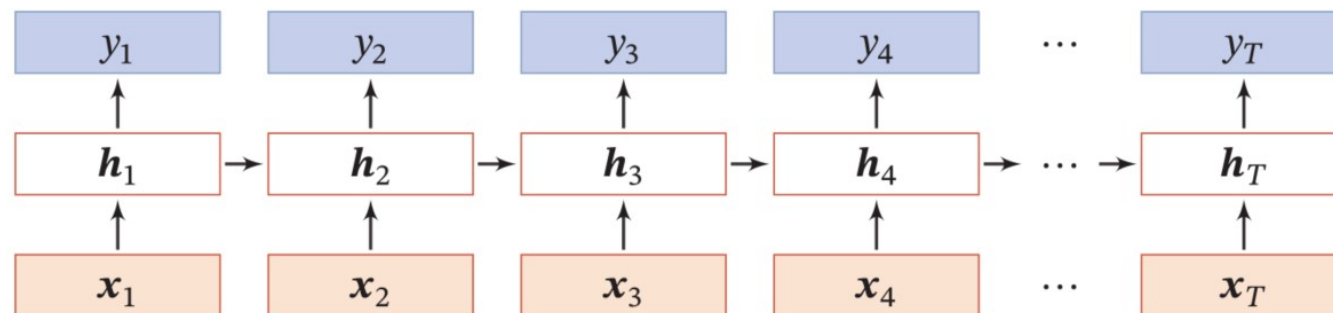
循环神经网络（ Recurrent Neural Network ， RNN ）

- ▶ 循环神经网络通过使用带自反馈的神经元，能够处理任意长度的时序数据。



- ▶ 循环神经网络比前馈神经网络更加符合生物神经网络的结构。
- ▶ 循环神经网络已经被广泛应用于语音识别、语言模型以及自然语言生成等任务上

按时间展开



简单循环网络（ Simple Recurrent Network ， SRN ）

► 状态更新：

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b})$$

► 一个完全连接的循环网络是任何非线性动力系统的近似器。

定理 6.1 – 循环神经网络的通用近似定理 [Haykin, 2009]： 如果一个完全连接的循环神经网络有足够数量的 sigmoid 型隐藏神经元，它可以以任意的准确率去近似任何一个非线性动力系统

$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, \mathbf{x}_t), \quad (6.10)$$

$$\mathbf{y}_t = o(\mathbf{s}_t), \quad (6.11)$$

其中 $\mathbf{s}_t$ 为每个时刻的隐状态， $\mathbf{x}_t$ 是外部输入， $g(\cdot)$ 是可测的状态转换函数， $o(\cdot)$ 是连续输出函数，并且对状态空间的紧致性没有限制。

图灵完备

- ▶ 图灵完备（Turing Completeness）是指一种数据操作规则，比如一种计算机编程语言，可以实现图灵机的所有功能，解决所有的可计算问题。

定理 6.2 – 图灵完备 [Siegelmann et al., 1991]: 所有的图灵机都可以被一个由使用 Sigmoid 型激活函数的神经元构成的全连接循环网络来进行模拟。

- ▶ 一个完全连接的循环神经网络可以近似解决所有的可计算问题。

循环神经网络

►作用

►输入 - 输出映射

- 机器学习模型（本节主要关注这种情况）

►存储器

- 联想记忆模型



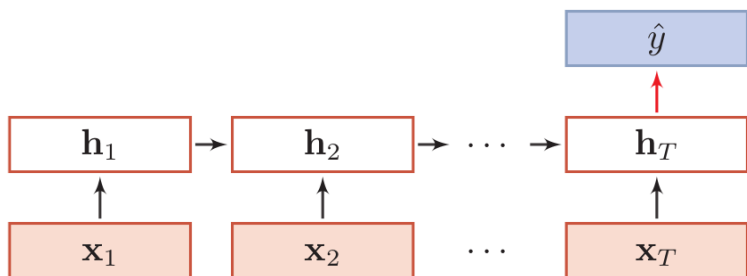
应用到机器学习

应用到机器学习

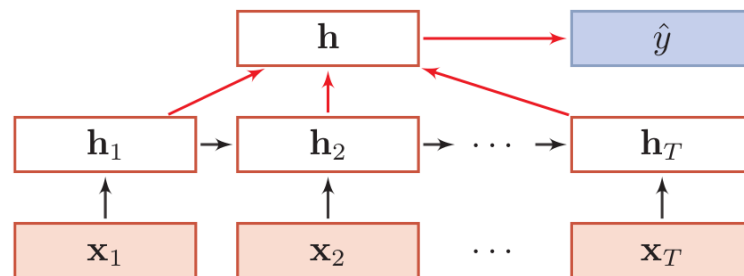
- ▶ 序列到类别
- ▶ 同步的序列到序列模式
- ▶ 异步的序列到序列模式

应用到机器学习

► 序列到类别



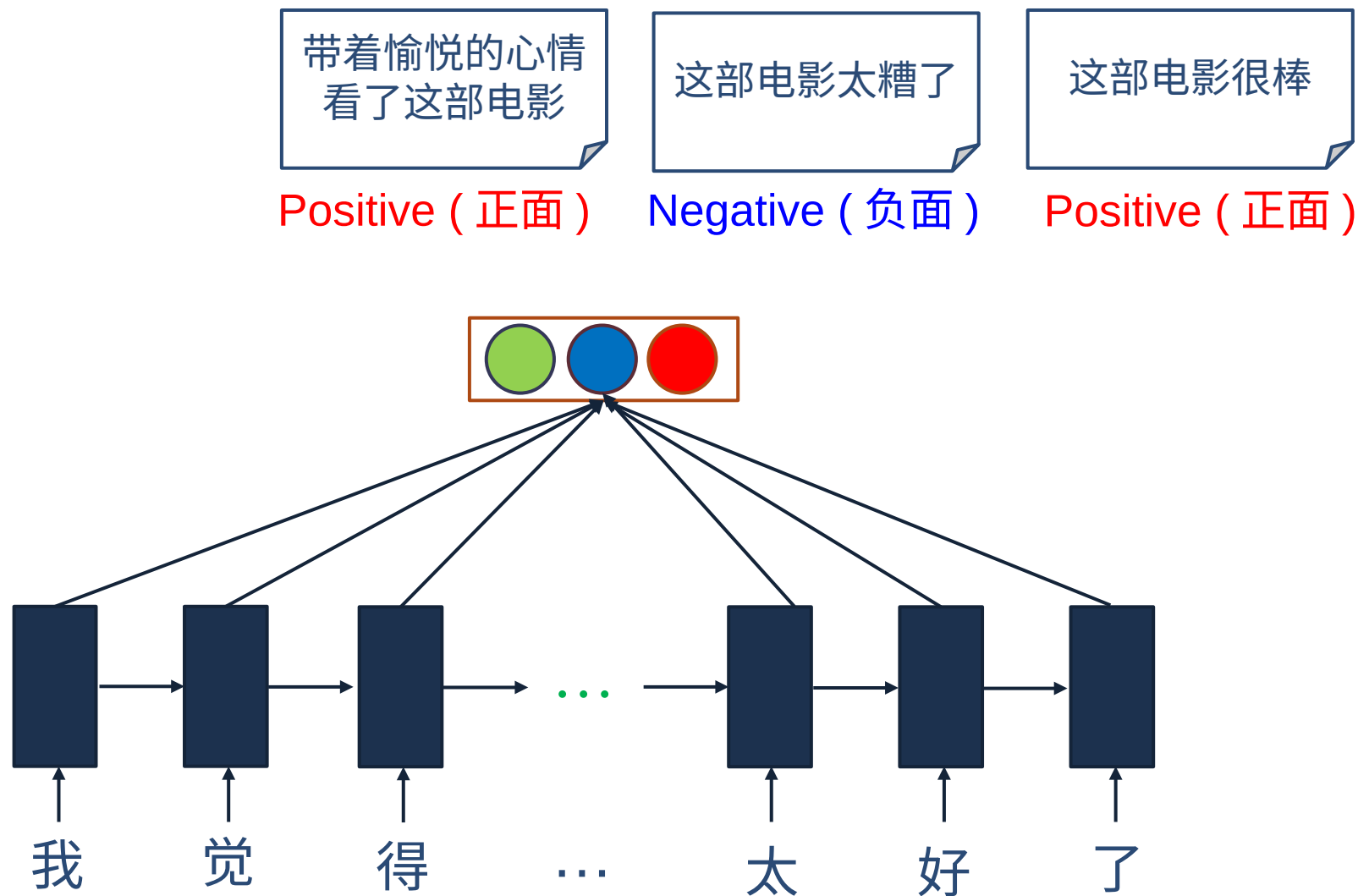
(a) 正常模式



(b) 按时间进行平均采样模式

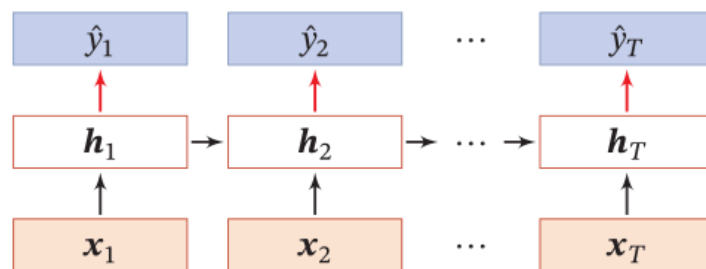
序列到类别

►情感分类



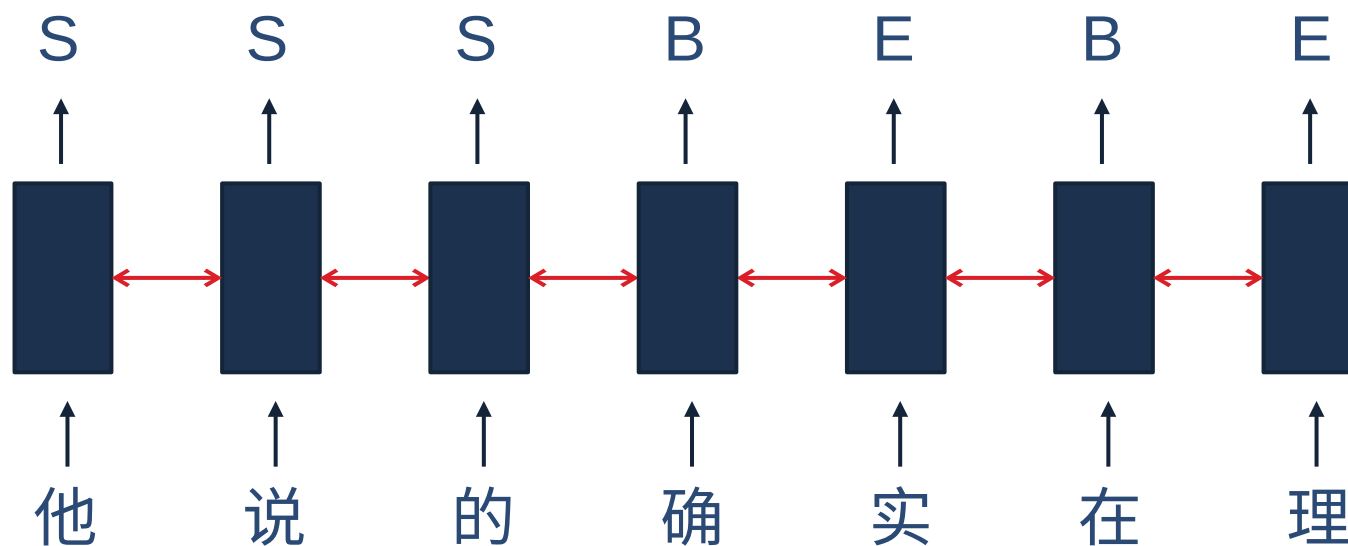
应用到机器学习

► 同步的序列到序列模式



同步的序列到序列模式

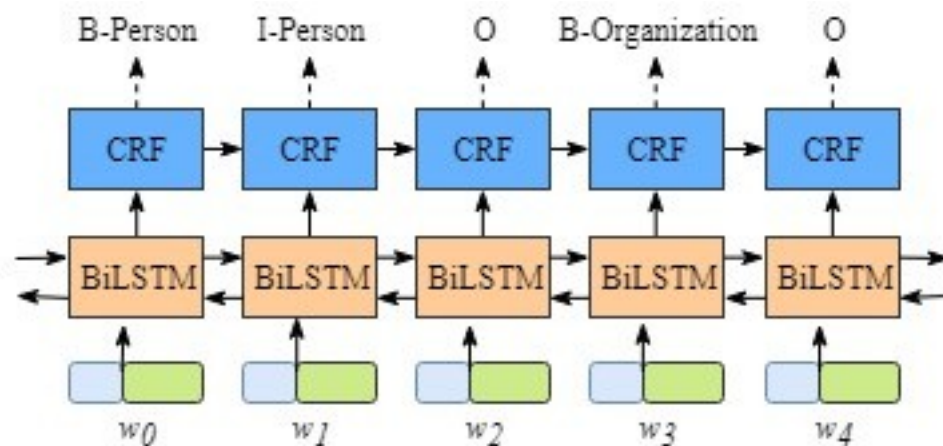
► 中文分词



同步的序列到序列模式

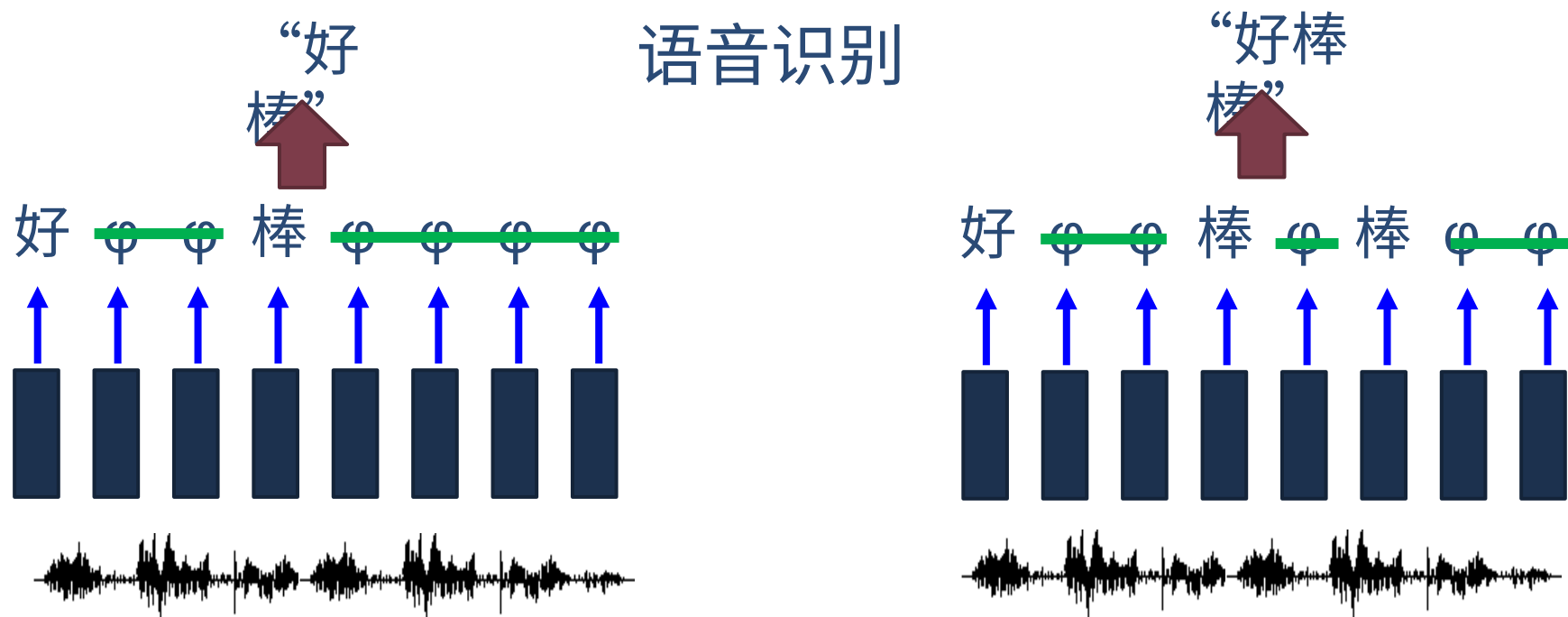
- 信息抽取 (Information Extraction , IE)
- 从无结构的文本中抽取结构化的信息，形成知识

小米创始人雷军表示，该公司 2015 年营收达到 780 亿元人民币，较 2014 年的 743 亿元人民币增长了 5%。



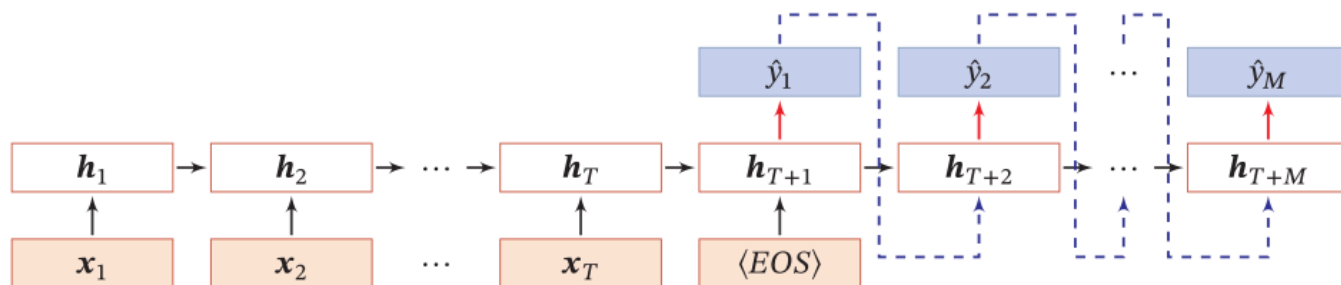
同步的序列到序列模式

- Connectionist Temporal Classification (CTC) [Alex Graves, ICML'06]
[Alex Graves, ICML'14][Haşim Sak, Interspeech'15][Jie Li, Interspeech'15][Andrew Senior, ASRU'15]



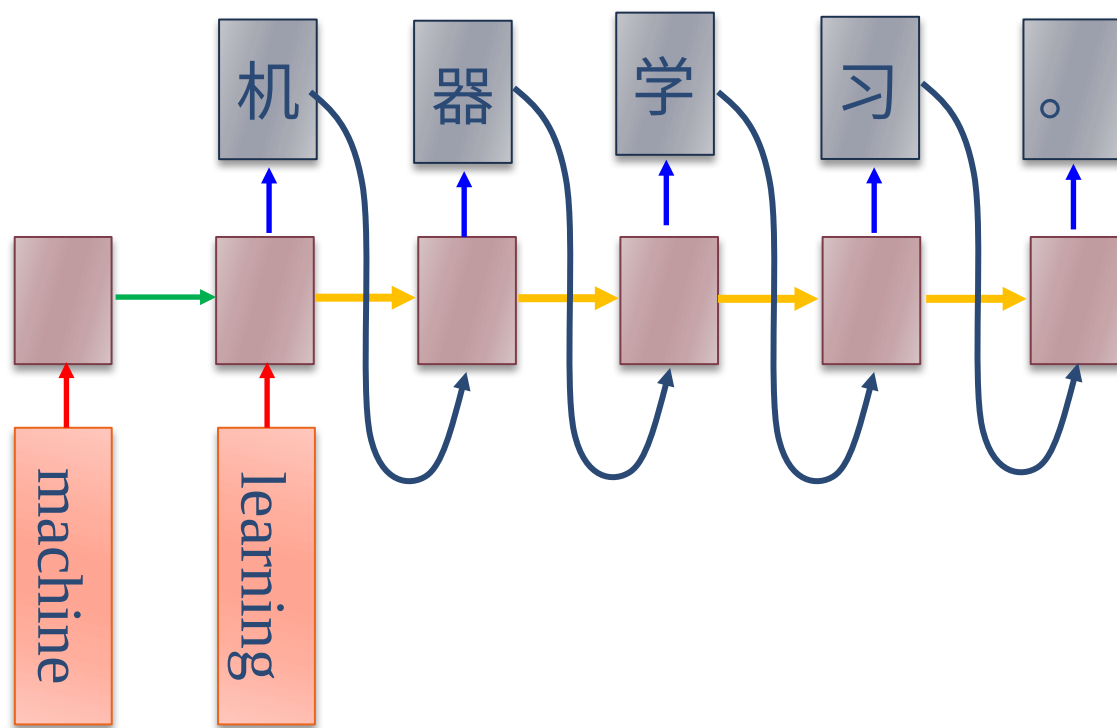
应用到机器学习

► 异步的序列到序列模式



异步的序列到序列模式

► 机器翻译



参数学习

► 机器学习

- 给定一个训练样本 $(\mathbf{x}, \mathbf{y})$ ，其中
 - $\mathbf{x} = (x_1, \dots, x_T)$ 为长度是 T 的输入序列，
 - $\mathbf{y} = (y_1, \dots, y_T)$ 是长度为 T 的标签序列。

- 时刻 t 的瞬时损失 $\mathcal{L}_t = \mathcal{L}(\mathbf{y}_t, g(\mathbf{h}_t))$,

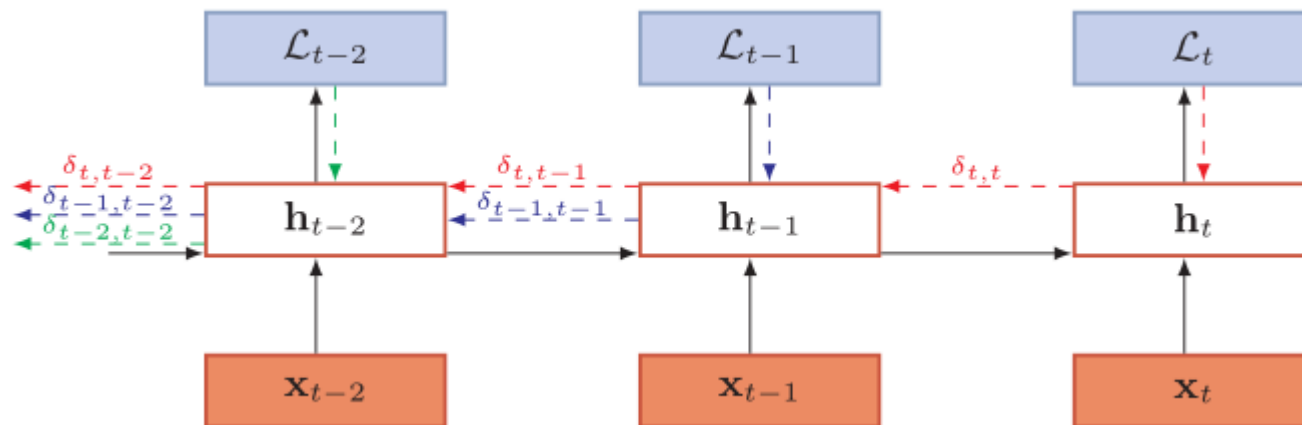
- 总损失函数

$$\mathcal{L} = \sum_{t=1}^T \mathcal{L}_t.$$

梯度

► 随时间反向传播算法

$$\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(U\mathbf{h}_t + W\mathbf{x}_{t+1} + \mathbf{b})$$



$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k} \mathbf{h}_{k-1}^T$$

$$\delta_{t,k} = \prod_{\tau=k}^{t-1} \left(\text{diag}(f'(\mathbf{z}_{\tau})) U^T \right) \delta_{t,t}$$

$\delta_{t,k}$ 为第 t 时刻的损失对第 k 步隐藏神经元的净输入 $\mathbf{z}_k$ 的导数

梯度消失 / 爆炸

► 梯度

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k} \mathbf{h}_{k-1}^T$$

► 其中

$$\delta_{t,k} = \prod_{\tau=k}^{t-1} \underbrace{\left(\text{diag}(f'(\mathbf{z}_{\tau})) U^T \right)}_{\lambda} \delta_{t,t}$$

由于梯度爆炸或消失问题，实际上只能学习到短周期的依赖关系。这就是所谓的长程依赖问题。

长程依赖问题

- ▶ 循环神经网络在时间维度上非常深！

- ▶ 梯度消失或梯度爆炸

- ▶ 如何改进？

- ▶ 梯度爆炸问题

- ▶ 权重衰减

- ▶ 梯度截断

- ▶ 梯度消失问题

- ▶ 改进模型

长程依赖问题

►改进方法

►循环边改为线性依赖关系

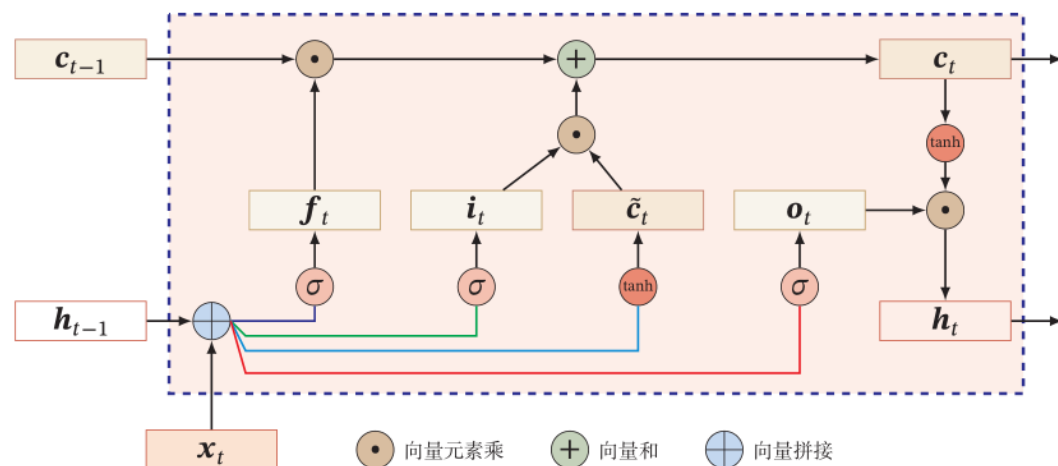
$$\mathbf{h}_t = \mathbf{h}_{t-1} + g(\mathbf{x}_t; \theta),$$

►增加非线性

$$\mathbf{h}_t = \mathbf{h}_{t-1} + g(\mathbf{x}_t, \mathbf{h}_{t-1}; \theta),$$

残差网络?

长短期记忆神经网络 (Long Short-Term Memory, LSTM)



$$\begin{aligned} \mathbf{i}_t &= \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i), \\ \mathbf{f}_t &= \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f), \\ \mathbf{o}_t &= \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o), \end{aligned}$$

$$\begin{aligned} \tilde{\mathbf{c}}_t &= \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \end{aligned}$$

LSTM 的各种变体

▶ 没有遗忘门

$$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t.$$

▶ 耦合输入门和遗忘 $\mathbf{f}_t + \mathbf{i}_t = \mathbf{1}$.

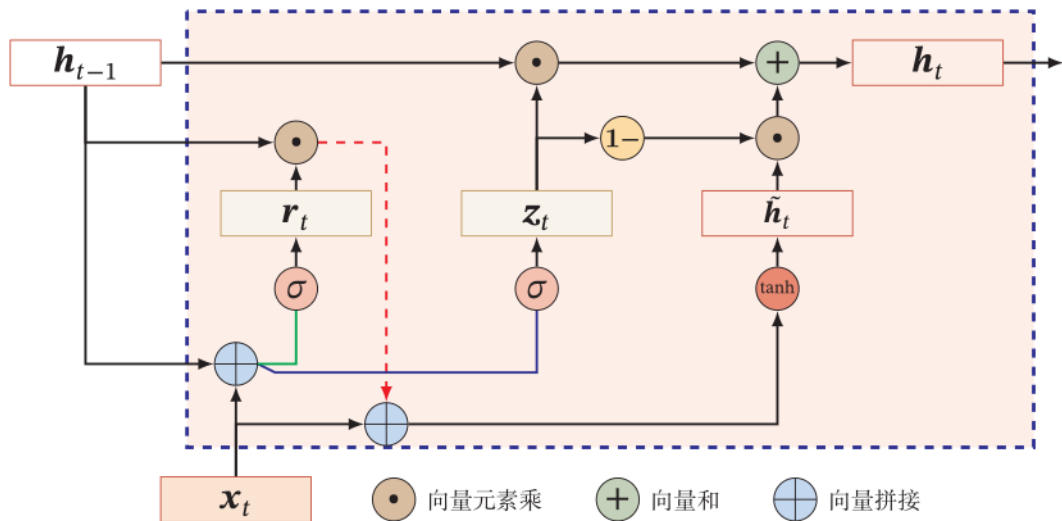
▶ peephole 连接

$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1} + \mathbf{b}_i),$$

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t + \mathbf{b}_o),$$

Gated Recurrent Unit, GRU



重置门

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r),$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z),$$

更新门

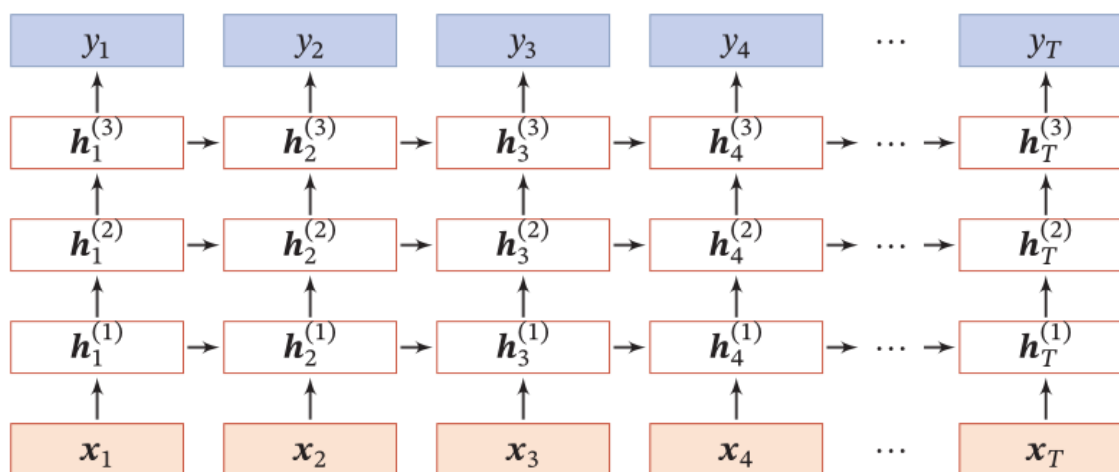
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t,$$

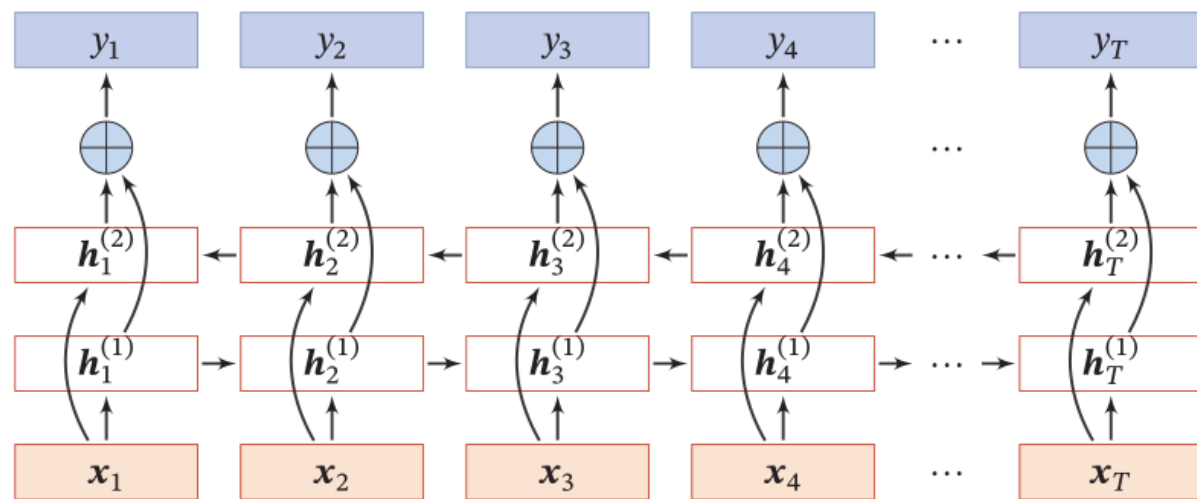


深层模型

堆叠循环神经网络



双向循环神经网络





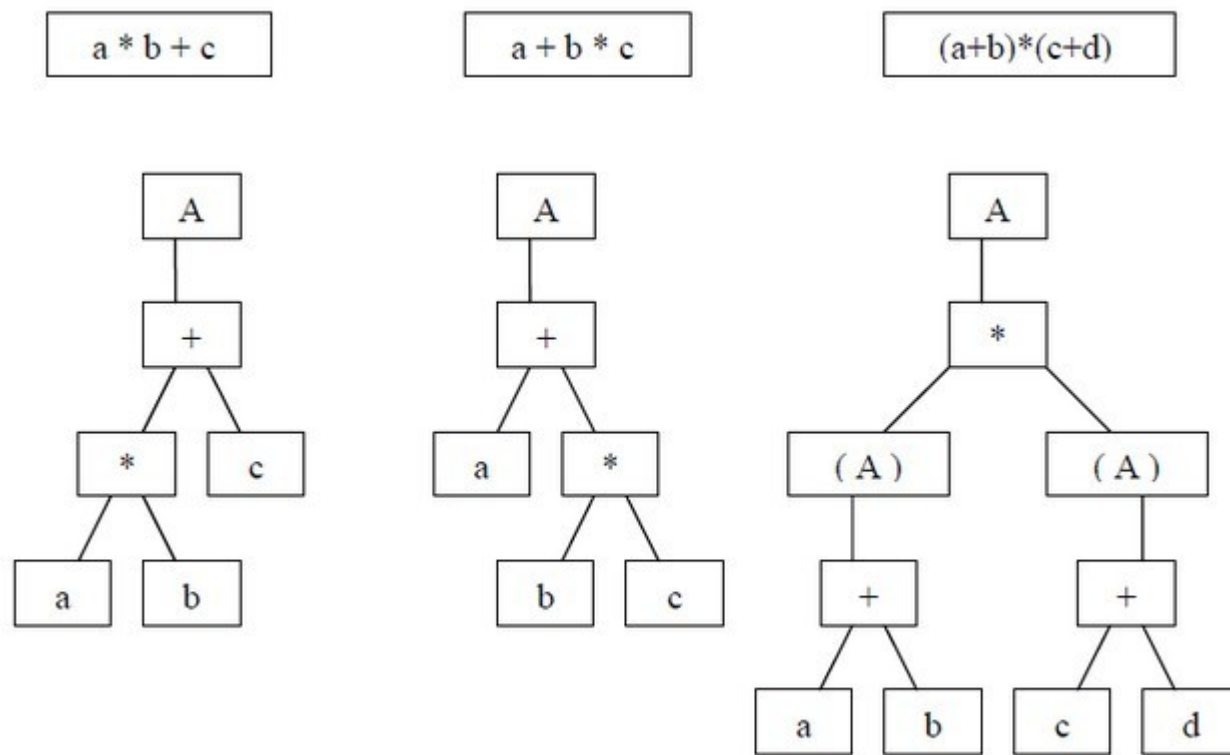
扩展到图结构

扩展到图结构



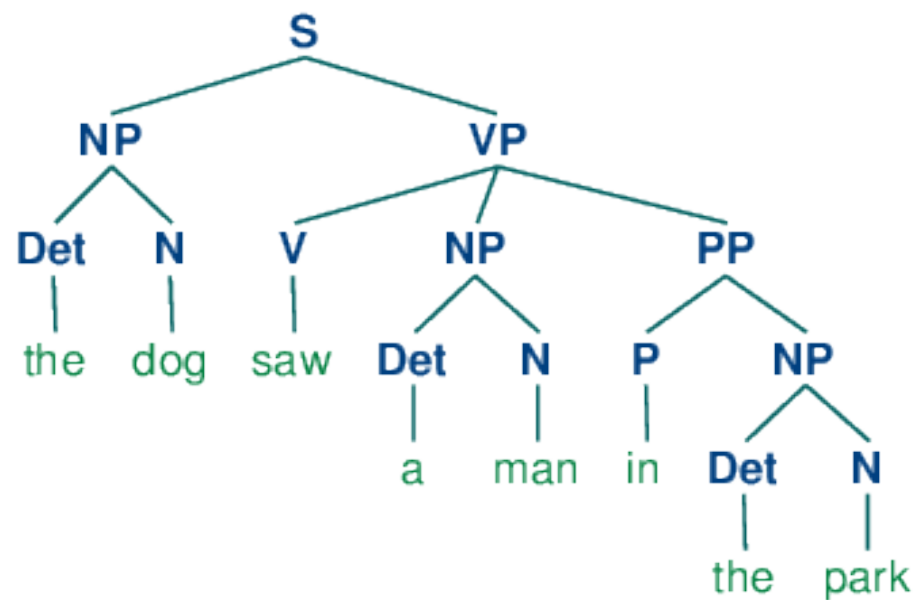
树结构

► 程序语言的句法结构

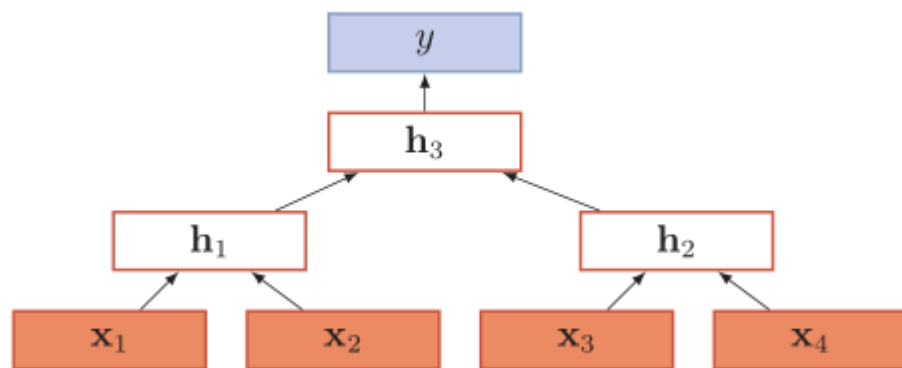


树结构

► 自然语言的句法结构



- 递归神经网络实在一个有向图无循环图上共享一个组合函数

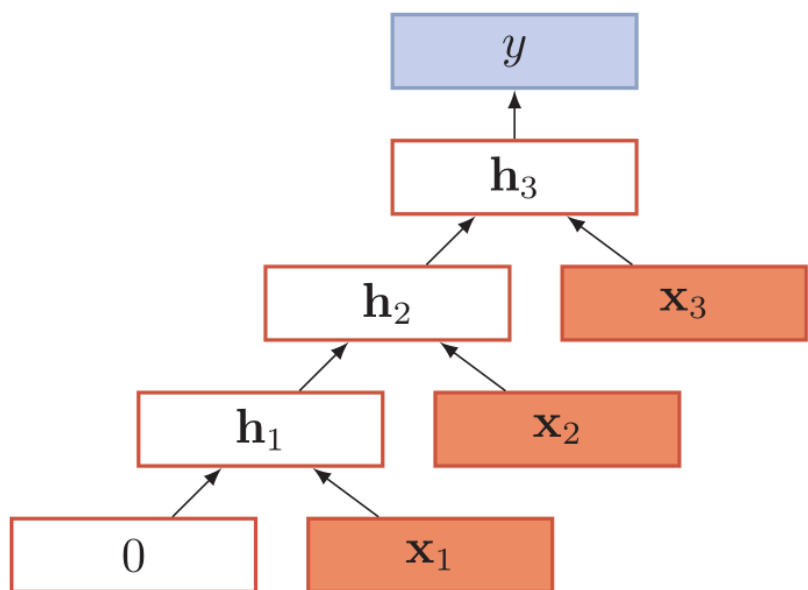


$$\mathbf{h}_i = f(\mathbf{h}_{\pi_i})$$

$$\begin{aligned}\mathbf{h}_1 &= f\left(W \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \mathbf{b}\right), \\ \mathbf{h}_2 &= f\left(W \begin{bmatrix} \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \mathbf{b}\right), \\ \mathbf{h}_3 &= f\left(W \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} + \mathbf{b}\right),\end{aligned}$$

递归神经网络

► 退化为循环神经网络



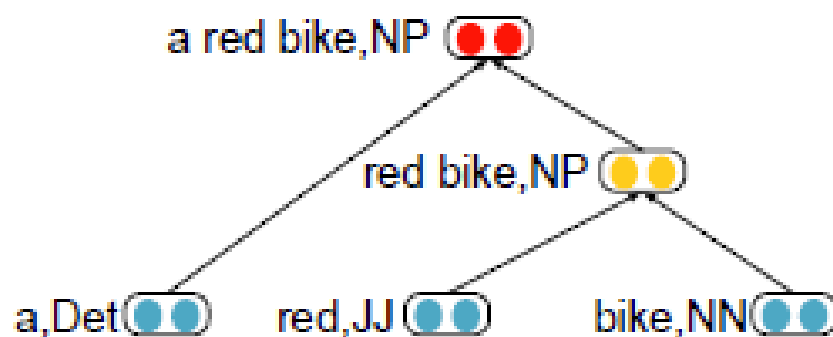
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t),$$

递归神经网络

给定一个语法树，

$p_2 \rightarrow ap_1$,

$p_1 \rightarrow bc$.



$$p_1 = f\left(W \begin{bmatrix} b \\ c \end{bmatrix}\right),$$
$$p_2 = f\left(W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right).$$

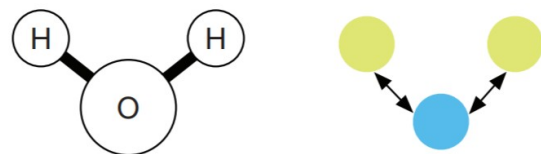
图网络

- ▶ 在实际应用中，很多数据是图结构的，比如知识图谱、社交网络、分子网络等。而前馈网络和循环网络很难处理图结构的数据。

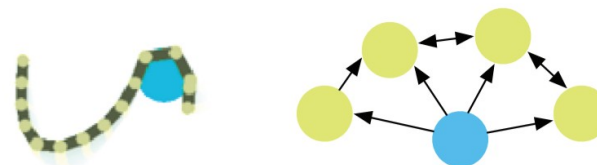
图数据

<https://arxiv.org/pdf/1806.01261.pdf>

(a) Molecule



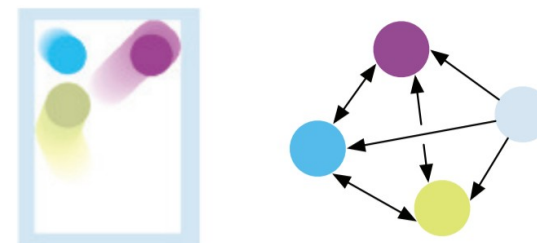
(b) Mass-Spring System



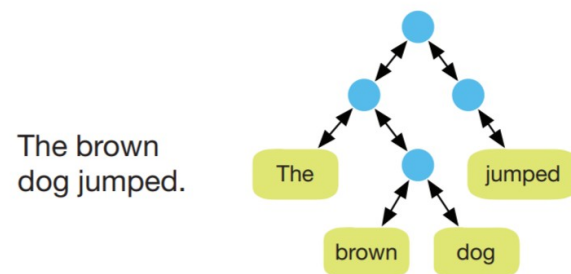
(c) n -body System



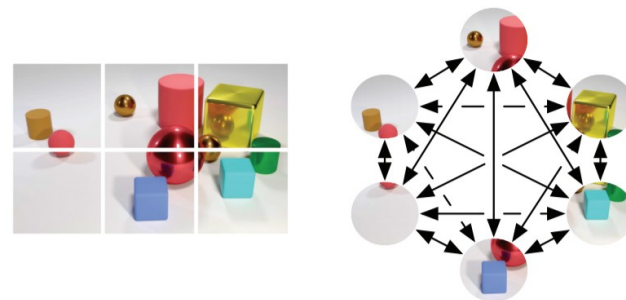
(d) Rigid Body System



(e) Sentence and Parse Tree

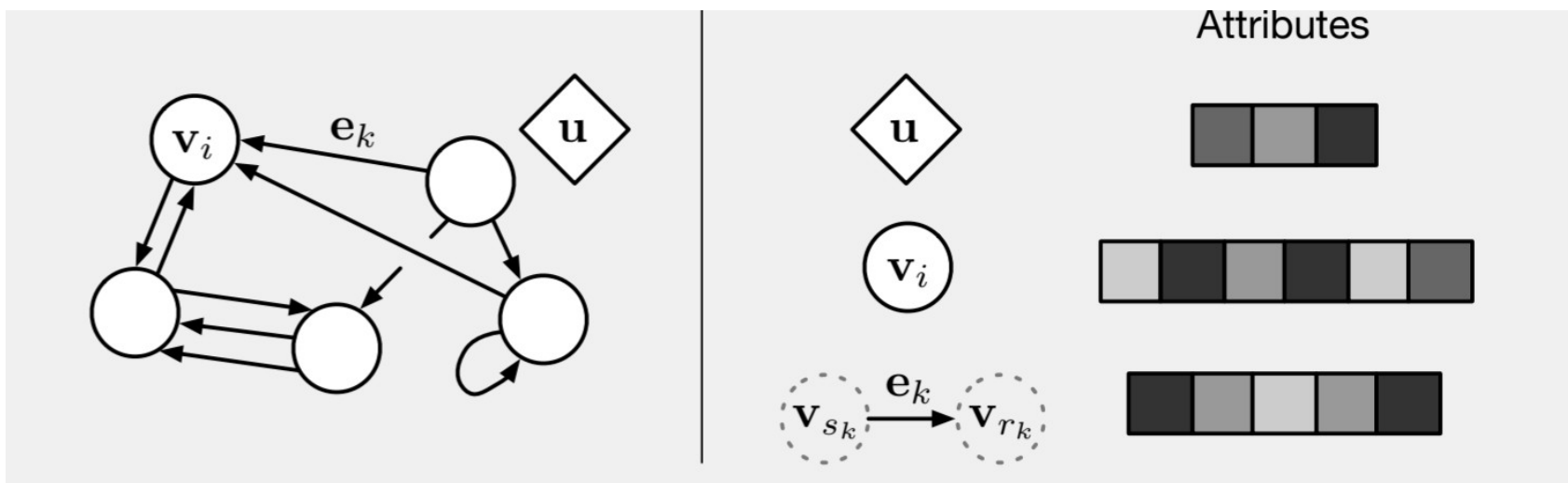


(f) Image and Fully-Connected Scene Graph



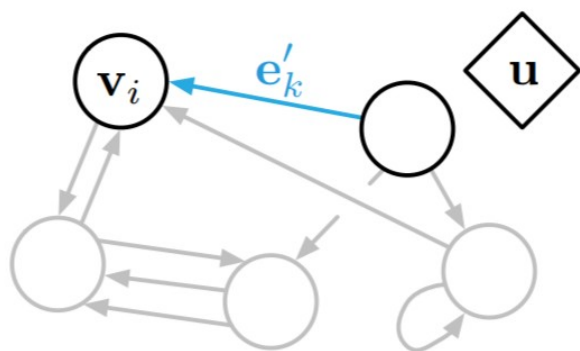
图网络

<https://arxiv.org/pdf/1806.01261.pdf>

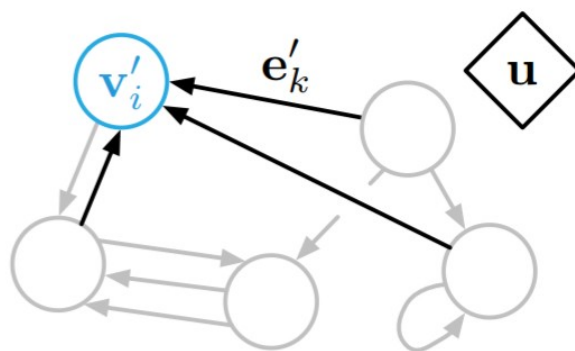


图网络

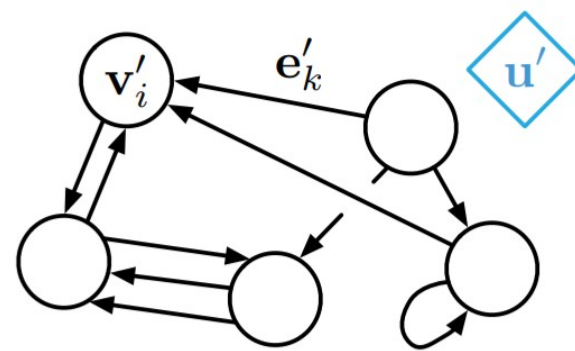
<https://arxiv.org/pdf/1806.01261.pdf>



(a) Edge update



(b) Node update



(c) Global update

图网络

► 对于一个任意的图结构 $G(V,E)$

► 更新函数

$$\mathbf{m}_t^{(v)} = \sum_{u \in N(v)} f(\mathbf{h}_{t-1}^{(v)}, \mathbf{h}_{t-1}^{(u)}, \mathbf{e}^{(u,v)})$$

$$\mathbf{h}_t^{(v)} = g(\mathbf{h}_{t-1}^{(v)}, \mathbf{m}_t^{(v)})$$

► 读出函数

$$\mathbf{y}_t = g(\{\mathbf{h}_T^{(v)} | v \in \mathcal{V}\})$$



循环网络应用

语言模型

► 自然语言理解 → 一个句子的可能性/合理性

► ! 在报那猫告做只



► 那只猫在作报告!



► 那个人在作报告!



► 一切都是概率!

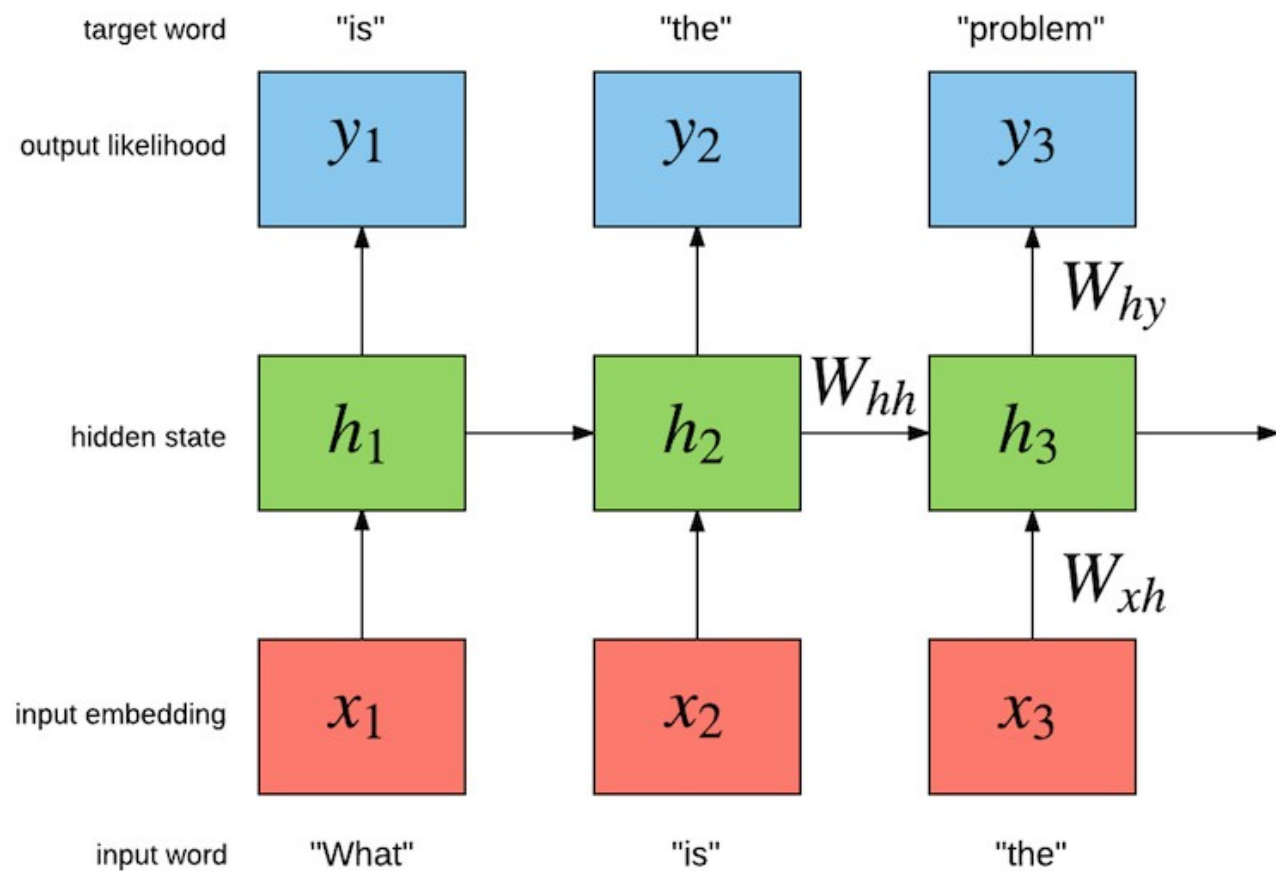
► $P(x_1, x_2, \dots, x_T)$

► $= \prod_i P(x_i | x_{i-1}, \dots, x_1)$

► $\approx \prod_i P(x_i | x_{i-1}, \dots, x_{i-n+1})$

N 元语言模
型

语言模型



生成 LINUX 内核代码

```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
    unsigned long flags;
    int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
    buf[0] = 0xFFFFFFFF & (bit << 4);
    min(inc, slist->bytes);
    printk(KERN_WARNING "Memory allocated %02x/%02x, "
        "original MLL instead\n"),
        min(min(multi_run - s->len, max) * num_data_in),
        frame_pos, sz + first_seg);
    div_u64_w(val, inb_p);
    spin_unlock(&disk->queue_lock);
    mutex_unlock(&s->sock->mutex);
    mutex_unlock(&func->mutex);
    return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
    if (tty == tty)
        disable_single_st_p(dev);
    pci_disable_spool(port);
}
```



作词机

► RNN 在 “学习” 过汪峰全部作品后自动生成的歌词

► <https://github.com/phunterlau/wangfeng-rnn>

我在这里中的夜里
就像一场是一种生命的意叶
就像我的生活变得在我一样
可我们这是一个知道
我只是一天你会怎吗
可我们这是我们是不要为你
我们想这有一种生活的时候

作诗

<p>白鹭窥鱼立， Egrets stood, peeping fishes. 青山照水开。 Water was still, reflecting mountains. 夜来风不动， The wind went down by nightfall, 明月见楼台。 as the moon came up by the tower.</p>	<p>满怀风月一枝春， Budding branches are full of romance. 未见梅花亦可人。 Plum blossoms are invisible but adorable. 不为东风无此客， With the east wind comes Spring. 世间何处是前身。 Where on earth do I come from?</p>
--	--

传统统计机器翻译

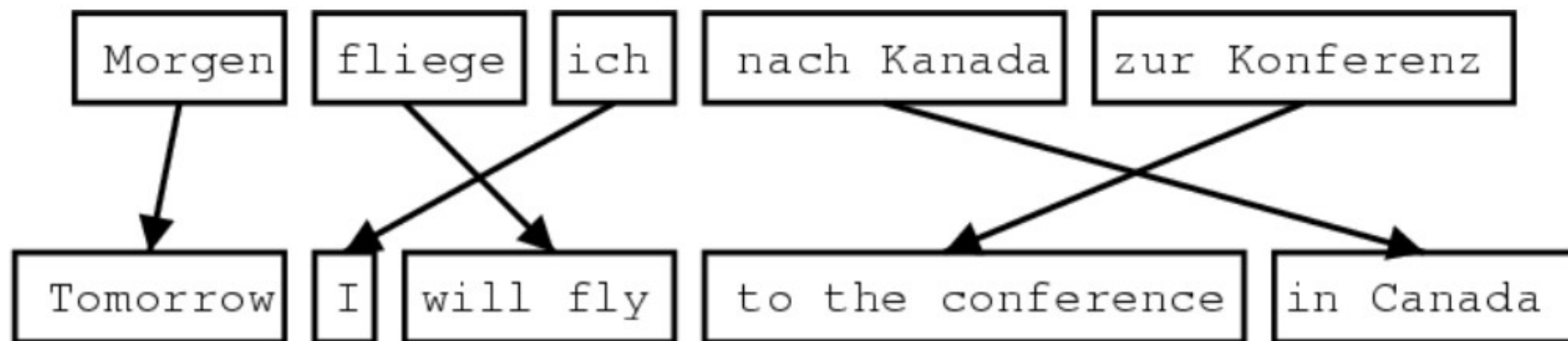
► 源语言: f

► 目标语言: e

► 模型: $\hat{e} = \operatorname{argmax}_e p(e|f) = \operatorname{argmax}_e p(f|e)p(e)$

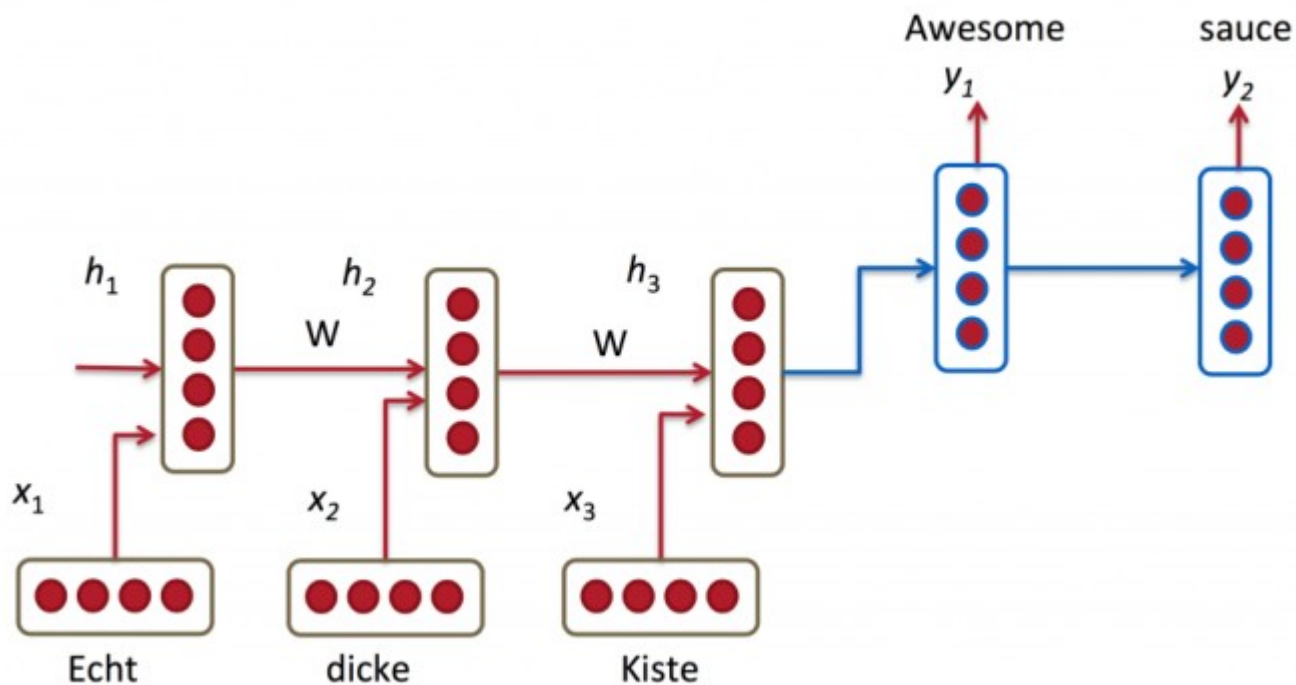
► $p(f|e)$: 翻译模型

► $p(e)$: 语言模型

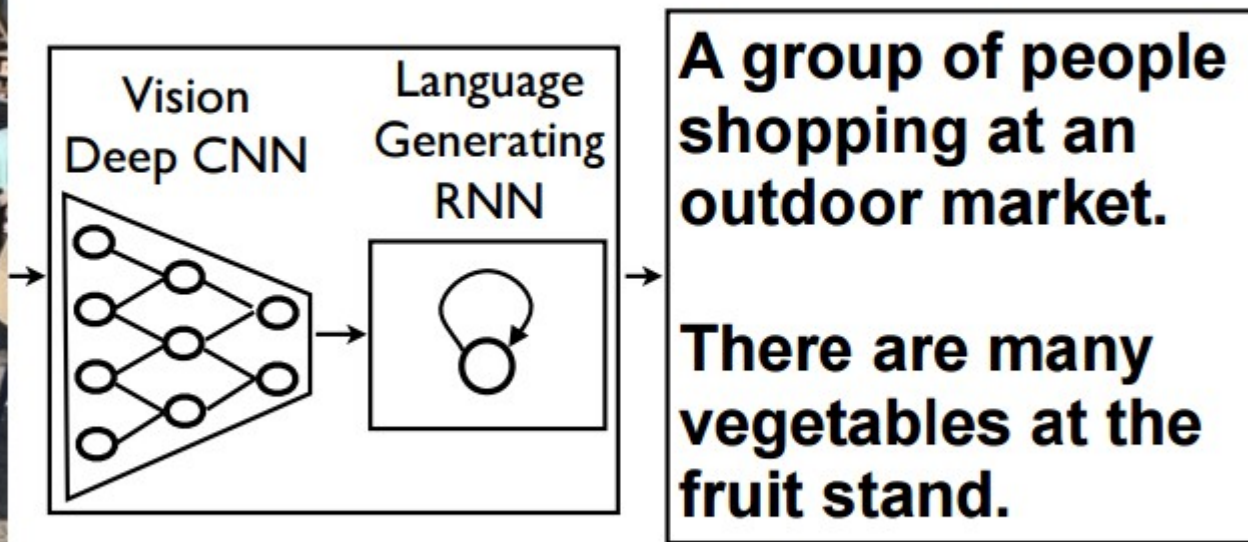


基于序列到序列的机器翻译

- ▶ 一个 RNN 用来编码
- ▶ 另一个 RNN 用来解码



看图说话



看图说话

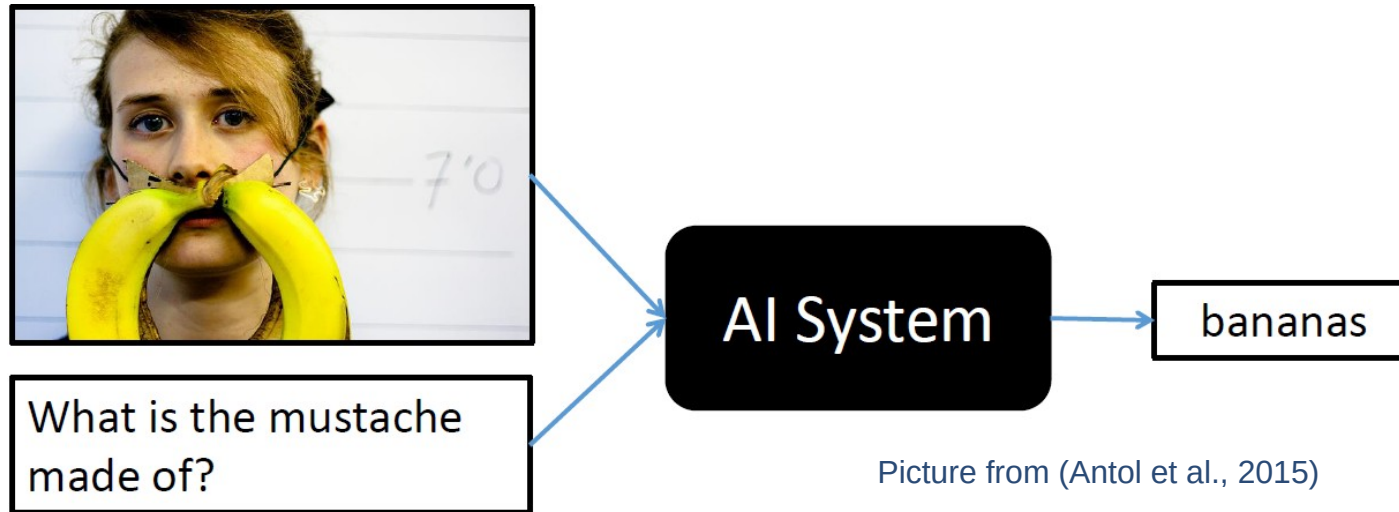


Figure 5. A selection of evaluation results, grouped by human rating.

Visual Question Answering (VQA)

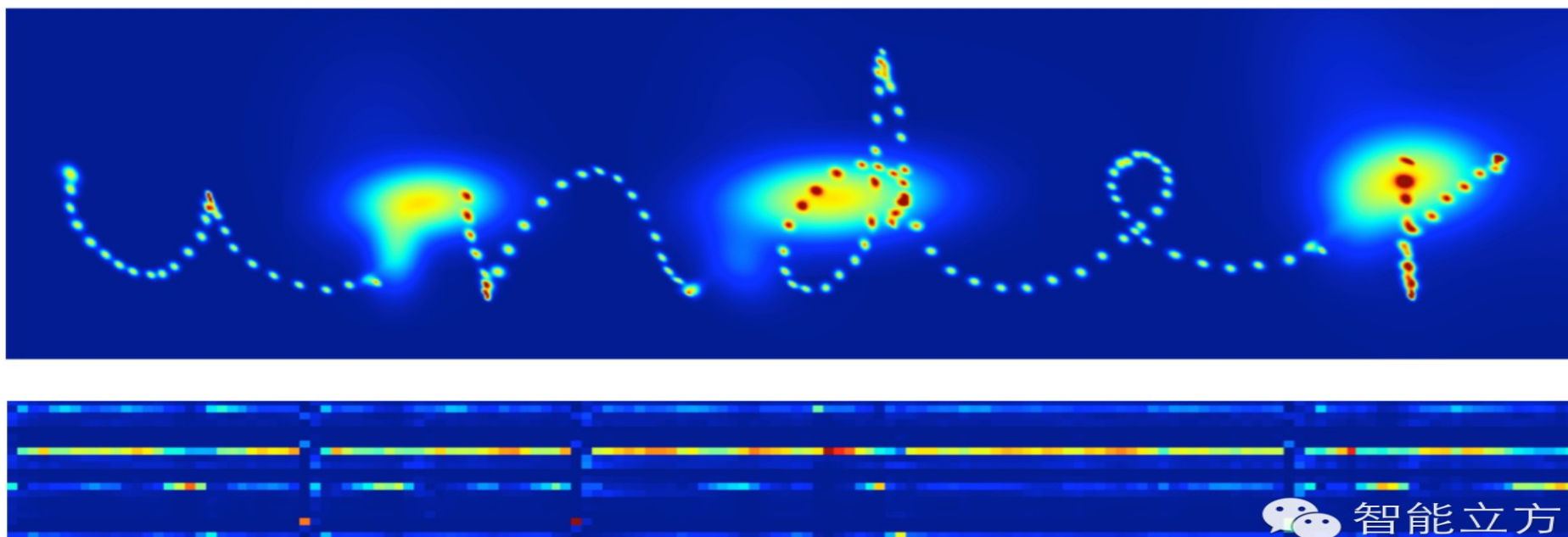
Demo Website

VQA: Given an image and a natural language question about the image, the task is to provide an accurate natural language answer



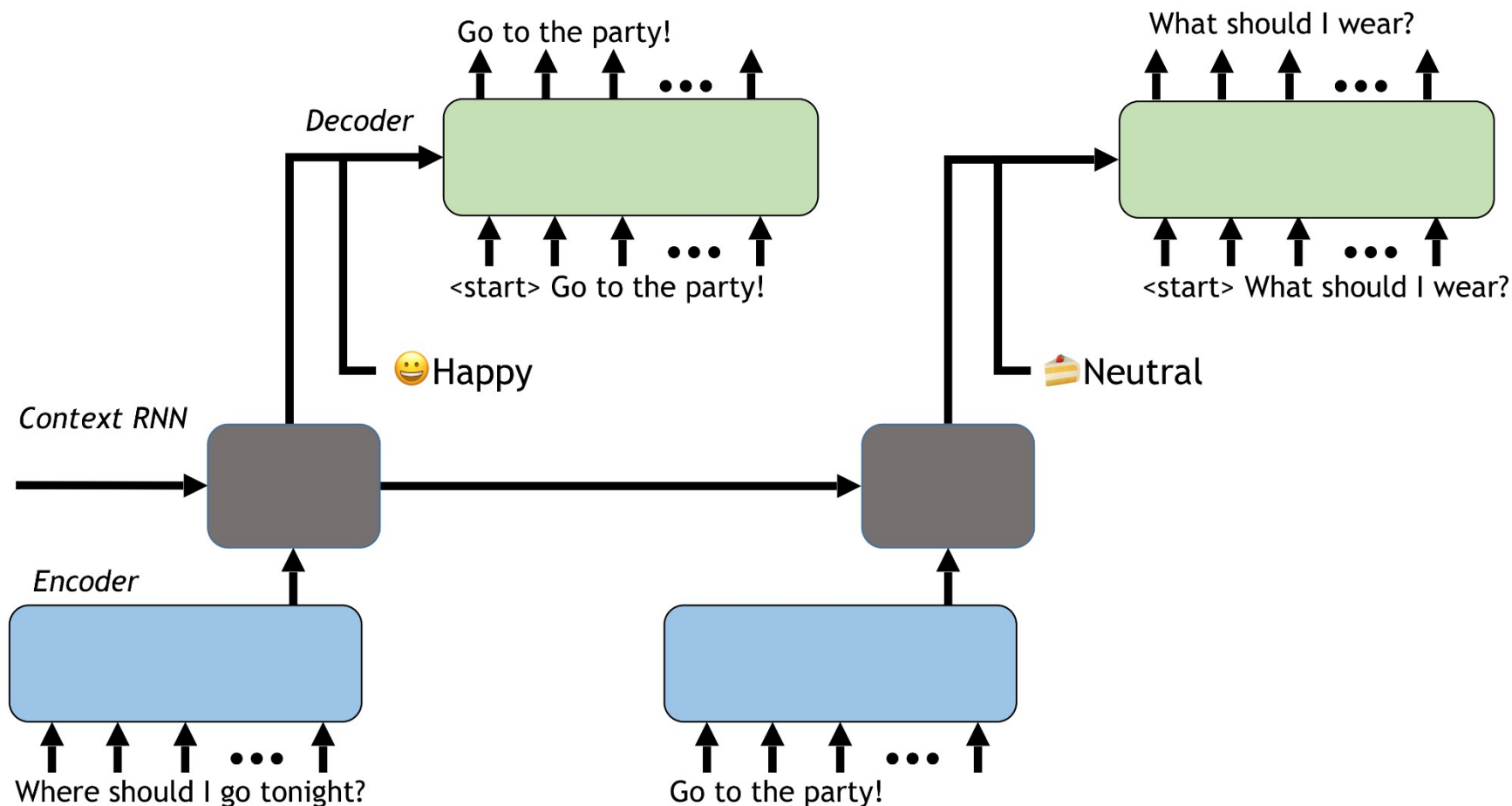
写字

- 把一个字母的书写轨迹看作是一连串的点。一个字母的“写法”其实是每一个点相对于前一个点的偏移量，记为 $(\text{offset } x, \text{offset } y)$ 。再增加一维取值为 0 或 1 来记录是否应该“提笔”。



对话系统

<https://github.com/lukalabs/cakechat>



循环神经网络总结

▶ 优点：

- ▶ 引入记忆
- ▶ 图灵完备

▶ 缺点：

- ▶ 长程依赖问题
- ▶ 记忆容量问题
- ▶ 并行能力

课后作业

► 编程练习

► <https://github.com/nndl/exercise/>

► [chap6 RNN](#)

- 1) 利用循环神经网络来生成唐诗
- 2) 利用循环神经网络来进行加法运算

谢 谢

<https://nndl.github.io/>