《神经网络与深度学习》

循环神经网络

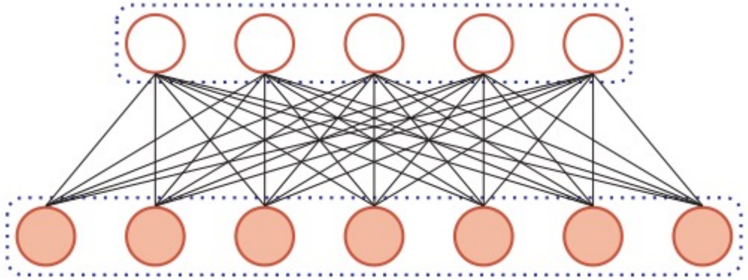
https://nndl.github.io/

参考资料

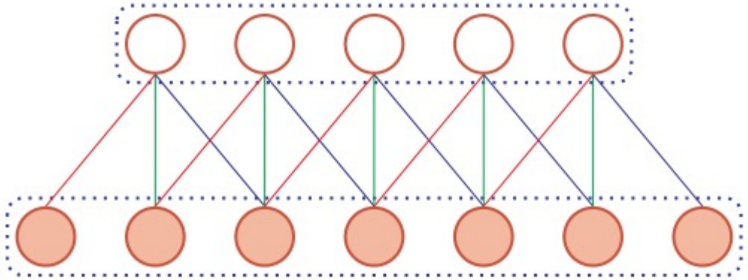
- 《神经网络与深度学习》第6章
  - https://nndl.github.io/
- 网络资料
  - An Introduction to Recurrent Neural Networks
    - https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912
  - Recurrent Neural Networks
    - https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

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。



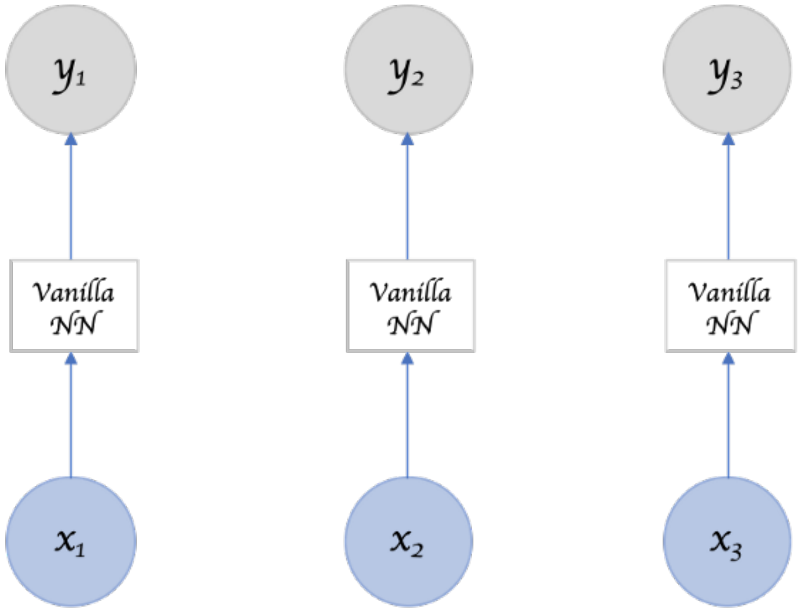
(a) 全连接层



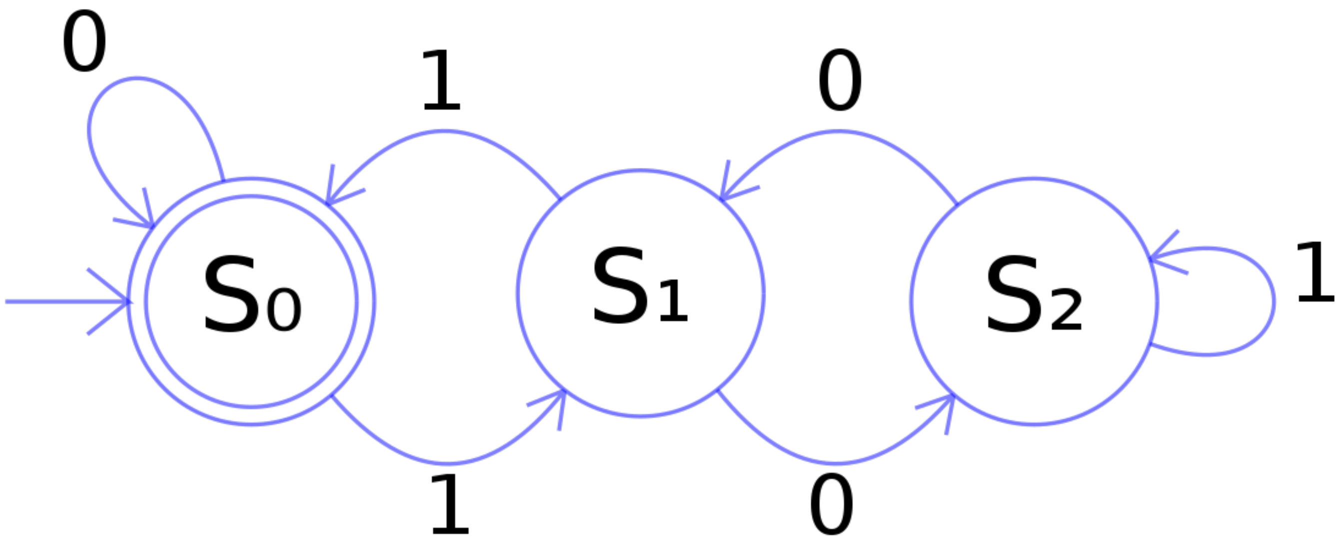
(b) 卷积层

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。



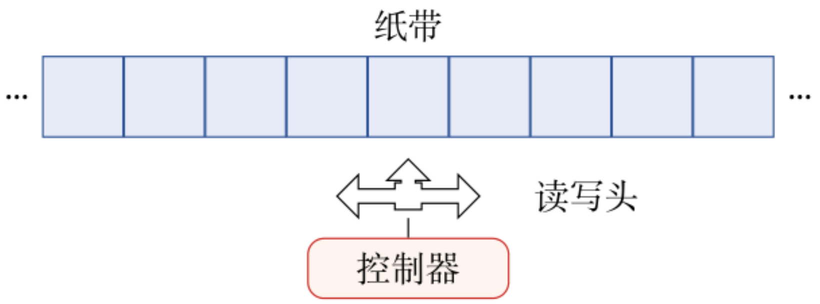
有限状态自动机（Finite Automata）



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

- 可计算问题
  - 函数 — $y = f(x)$
  - 有限状态机
  - 图灵机

前馈网络

通用近似定理

需要记忆能力

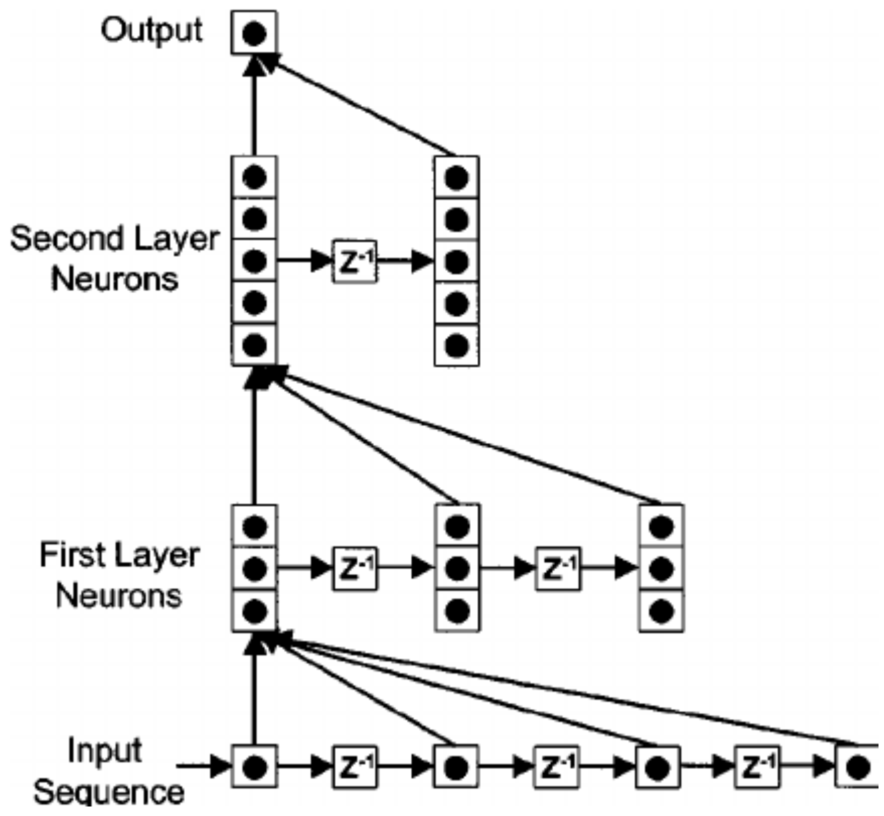
如何给网络增加记忆能力？

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）

$$\boldsymbol{h}_t^{(l)} = f\left(\boldsymbol{h}_t^{(l-1)}, \boldsymbol{h}_{t-1}^{(l-1)}, \cdots, \boldsymbol{h}_{t-K}^{(l-1)}\right)$$

  - 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，用变量y_t的历史信息来预测自己

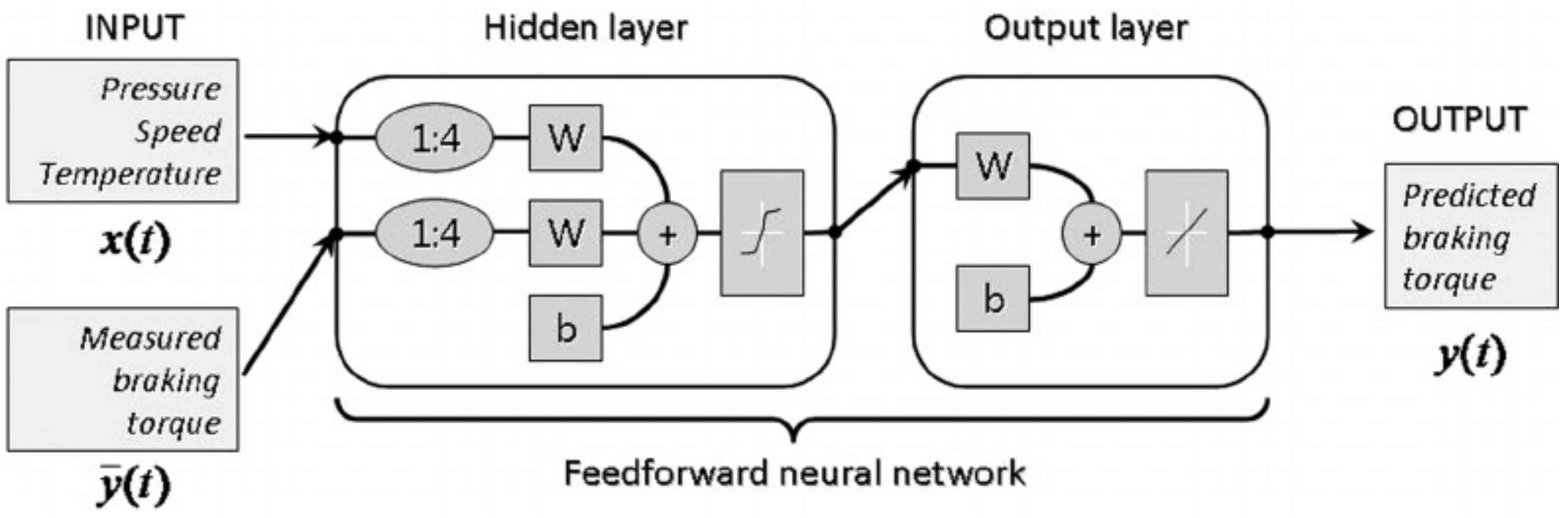
$$\boldsymbol{y}_t = w_0 + \sum_{k=1}^{K} w_k \boldsymbol{y}_{t-k} + \epsilon_t$$

  - $\epsilon_t \sim N(0, \sigma^2)$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）

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

  - 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



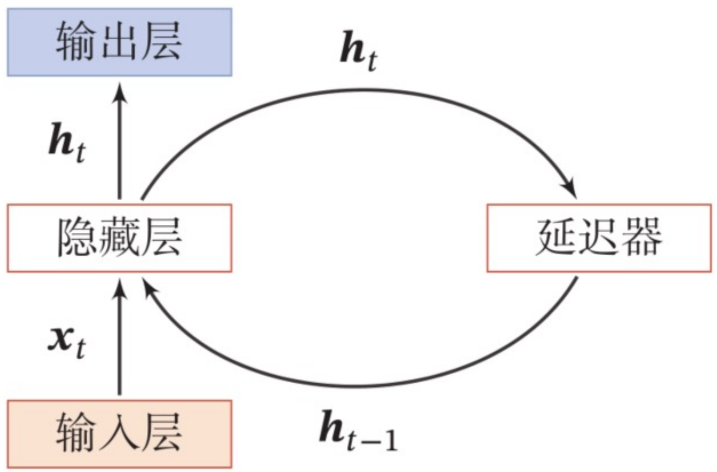
https://www.researchgate.net/publication/234052442_Braking_torque_control_using_reccurent_neural_networks

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。

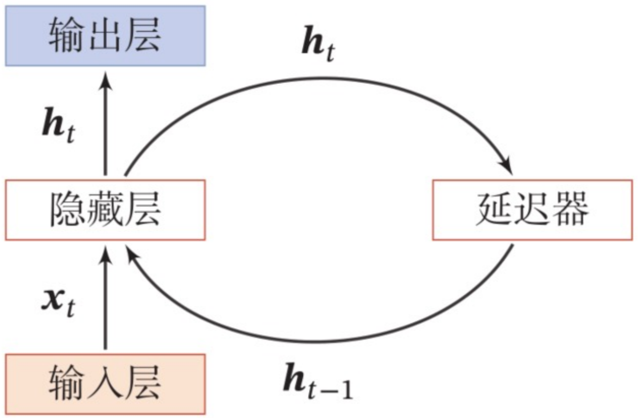
$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

活性值
状态

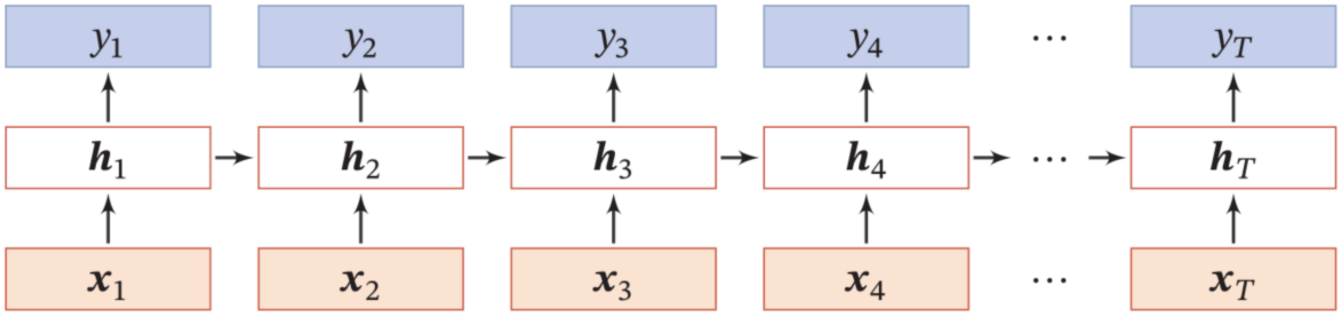


- 循环神经网络比前馈神经网络更加符合生物神经网络的结构。
- 循环神经网络已经被广泛应用在语音识别、语言模型以及自然语言生成等任务上

按时间展开



输出层
h_t
h_t
隐藏层
延迟器
x_t
输入层
h_{t-1}



y_1 y_2 y_3 y_4 ⋯ y_T
h_1 h_2 h_3 h_4 ⋯ h_T
x_1 x_2 x_3 x_4 ⋯ x_T

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

- 状态更新：

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

- 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

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

$$\boldsymbol{y}_t = o(\boldsymbol{s}_t), \tag{6.11}$$

其中 $\boldsymbol{s}_t$ 为每个时刻的隐状态，$\boldsymbol{x}_t$ 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

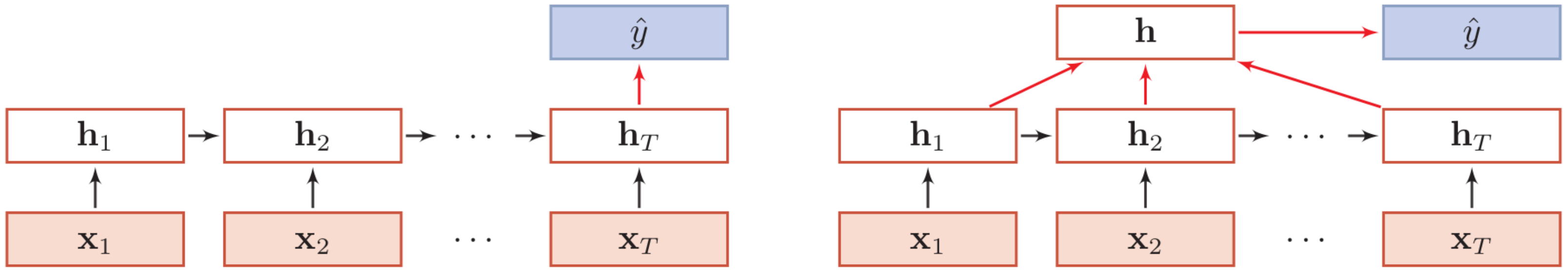
应用到机器学习

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

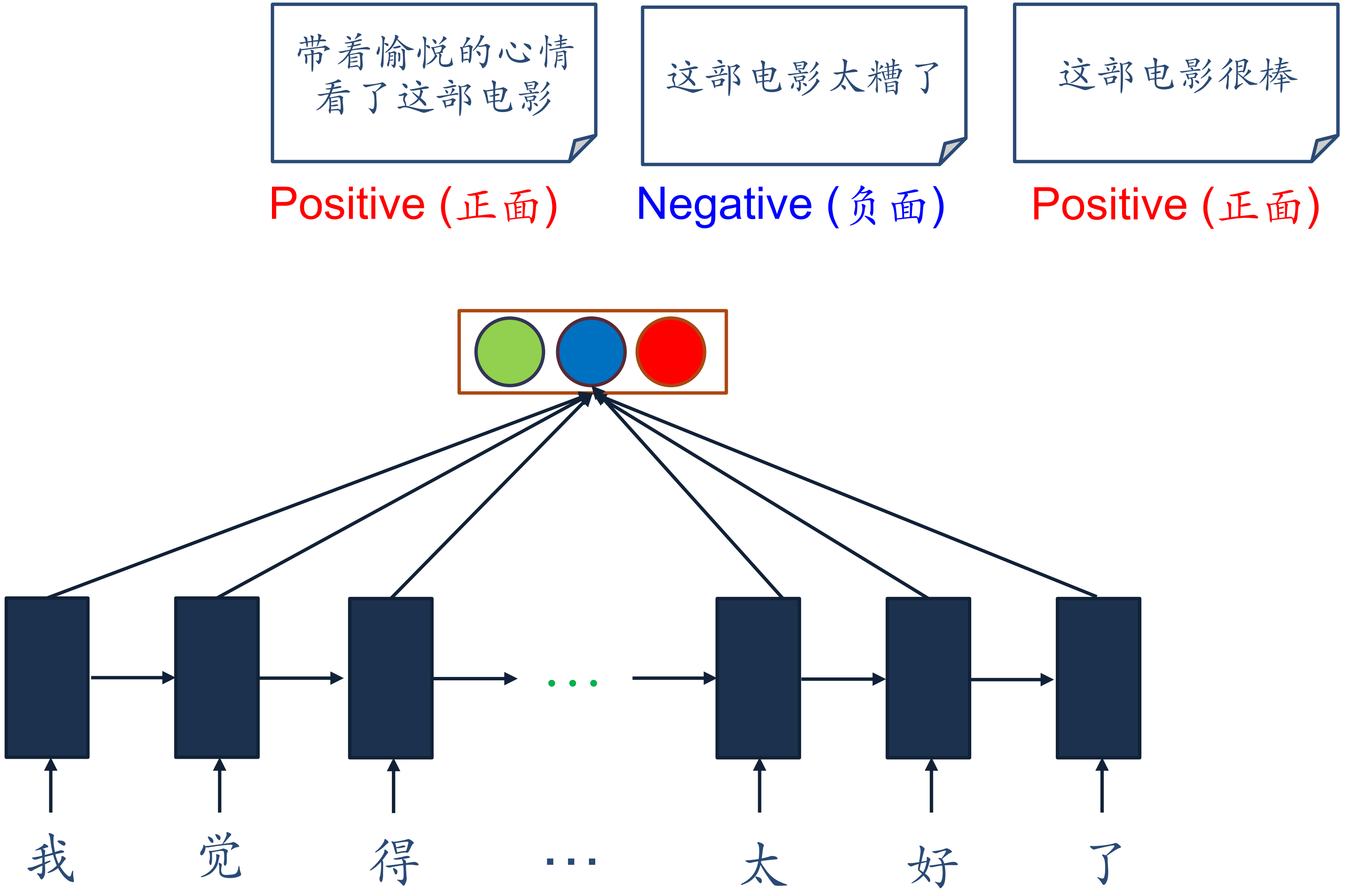
(b) 按时间进行平均采样模式

$$\hat{y} = g(h_T)$$

$$\hat{y} = g\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right)$$

序列到类别

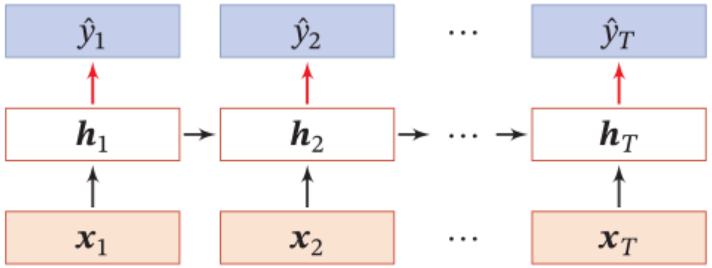
- 情感分类



带着愉悦的心情看了这部电影
这部电影太糟了
这部电影很棒
Positive (正面)
Negative (负面)
Positive (正面)
我
觉
得
…
太
好
了

应用到机器学习

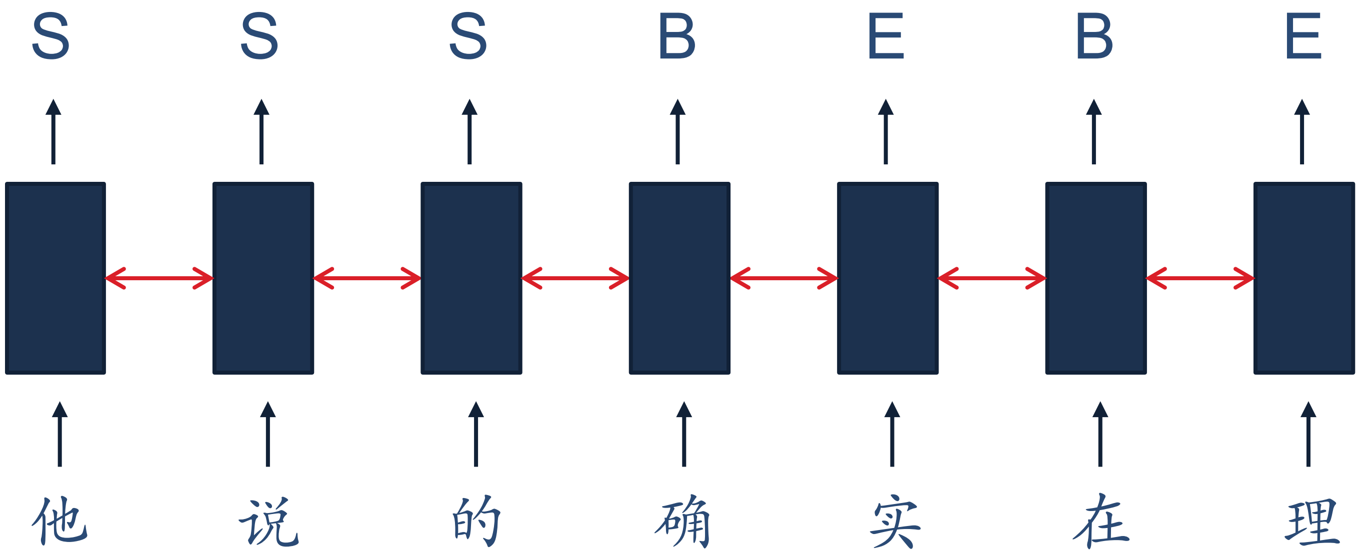
- 同步的序列到序列模式



$$\hat{y}_t = g(h_t), \forall t \in [1, T]$$

同步的序列到序列模式

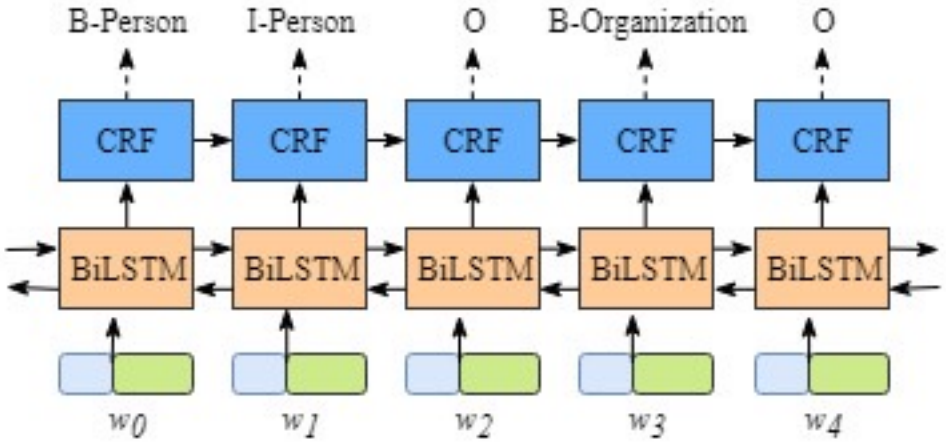
- 中文分词



同步的序列到序列模式

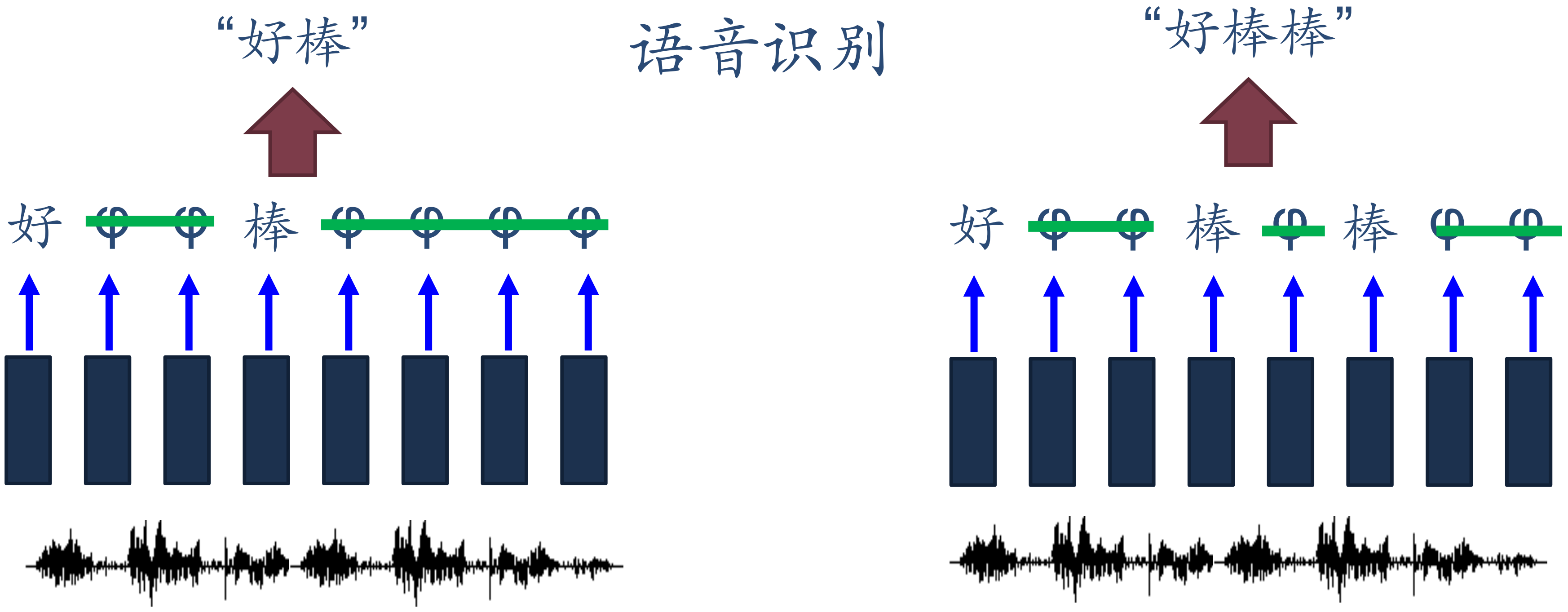
- 信息抽取(Information Extraction，IE)
  - 从无结构的文本中抽取结构化的信息，形成知识

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%。



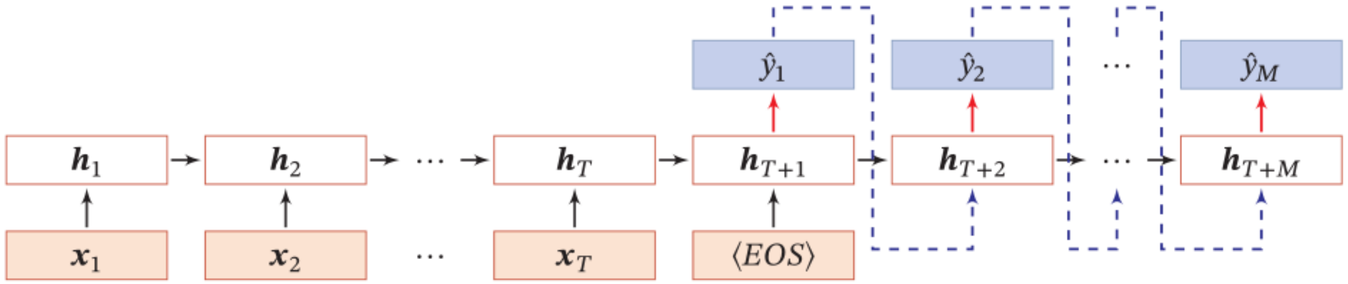
同步的序列到序列模式

- 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

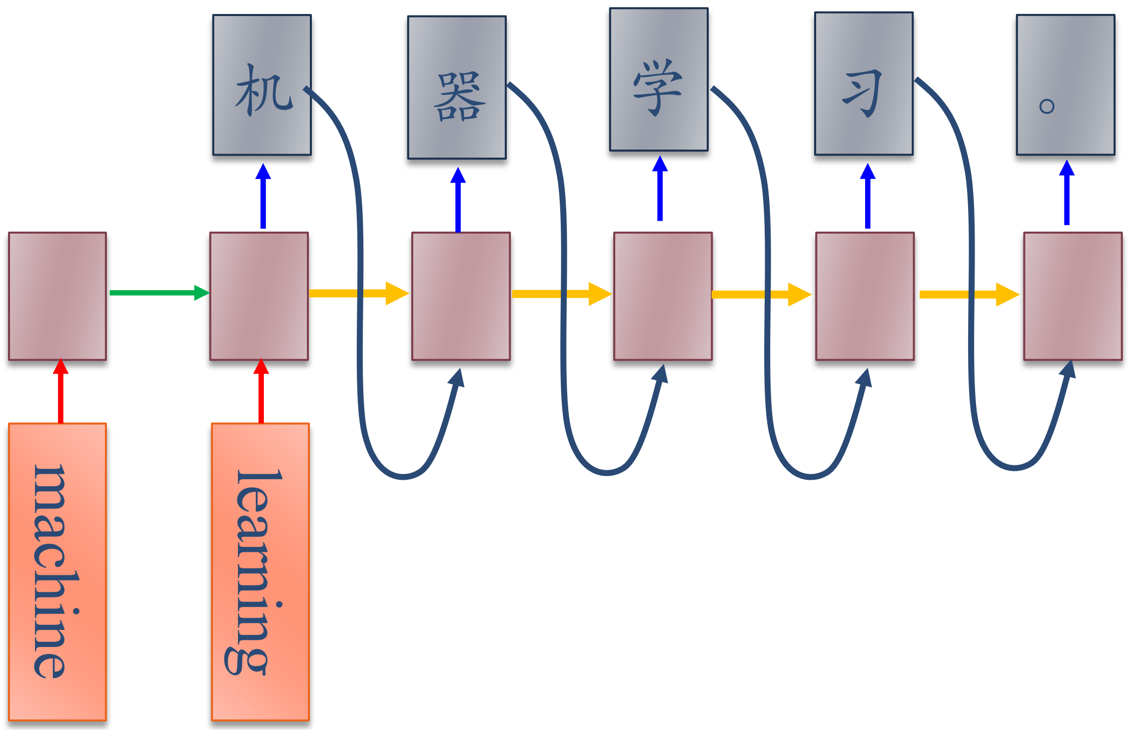
- 异步的序列到序列模式



$$
\begin{aligned}
h_t &= f_1(h_{t-1}, x_t), \forall t \in [1, T] \\
h_{T+t} &= f_2(h_{T+t-1}, \hat{y}_{t-1}), \forall t \in [1, M] \\
\hat{y}_t &= g(h_{T+t}), \forall t \in [1, M]
\end{aligned}
$$

异步的序列到序列模式

- 机器翻译



参数学习

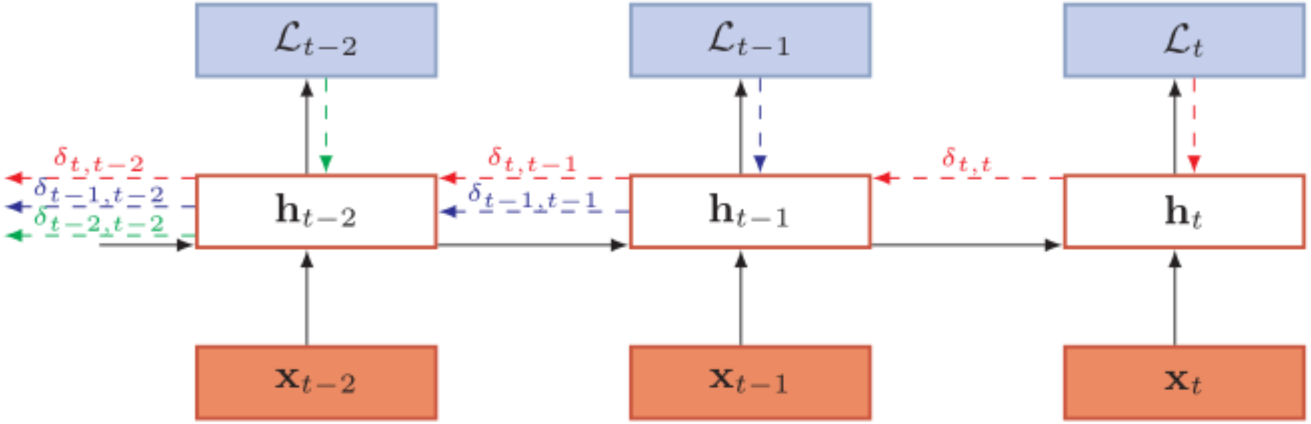
- 机器学习
  - 给定一个训练样本(x,y)，其中
    - x = (x1 ,… ,xT)为长度是T 的输入序列，
    - y = (y1 ,… ,yT)是长度为T 的标签序列。
- 时刻t的瞬时损失函数为 $\mathcal{L}_t = \mathcal{L}(\mathbf{y}_t, g(\mathbf{h}_t)),$
- 总损失函数 $\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}_t.$

梯度推导，How？

梯度

- 随时间反向传播算法

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



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

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

$\delta_{t,k}$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

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

- 其中

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

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

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

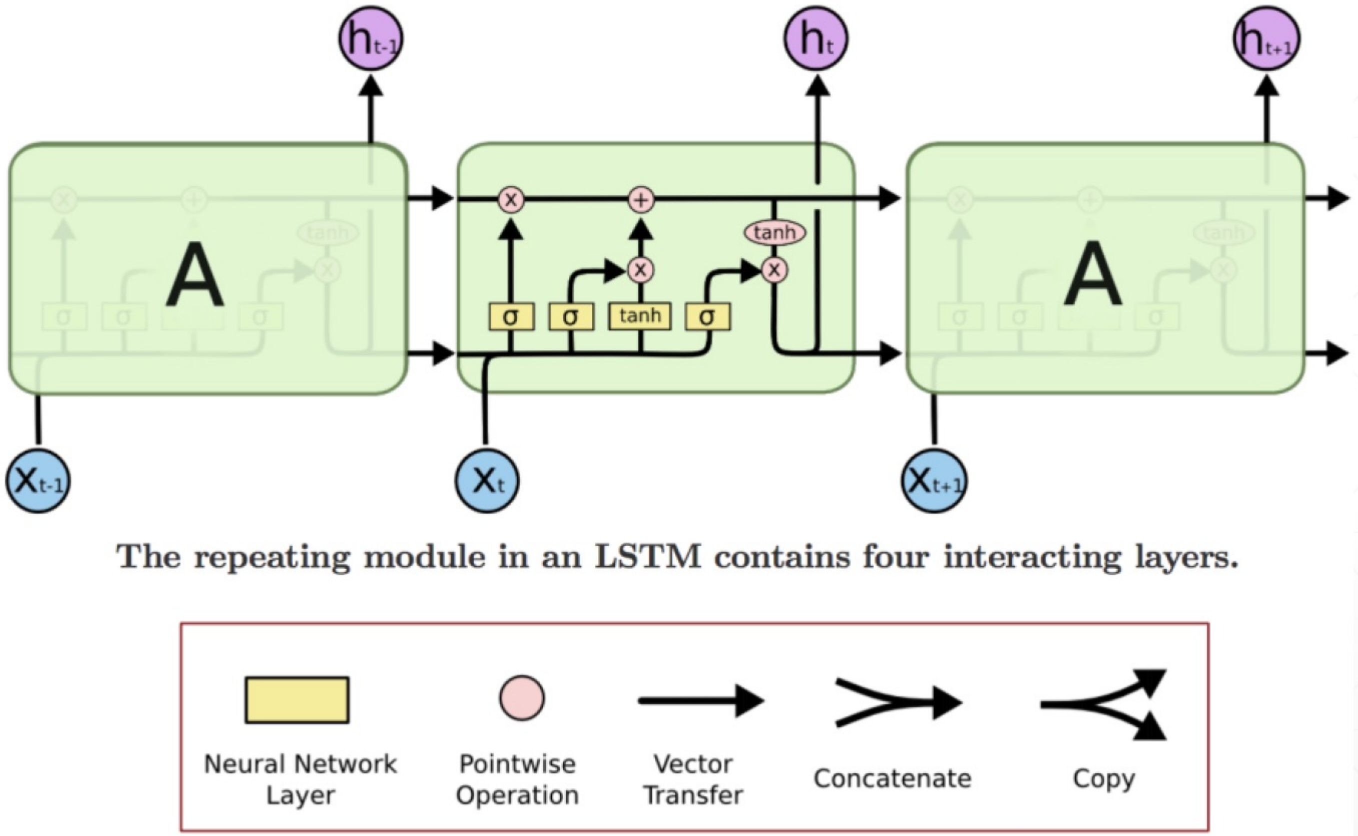
  - 增加非线性

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

残差网络?

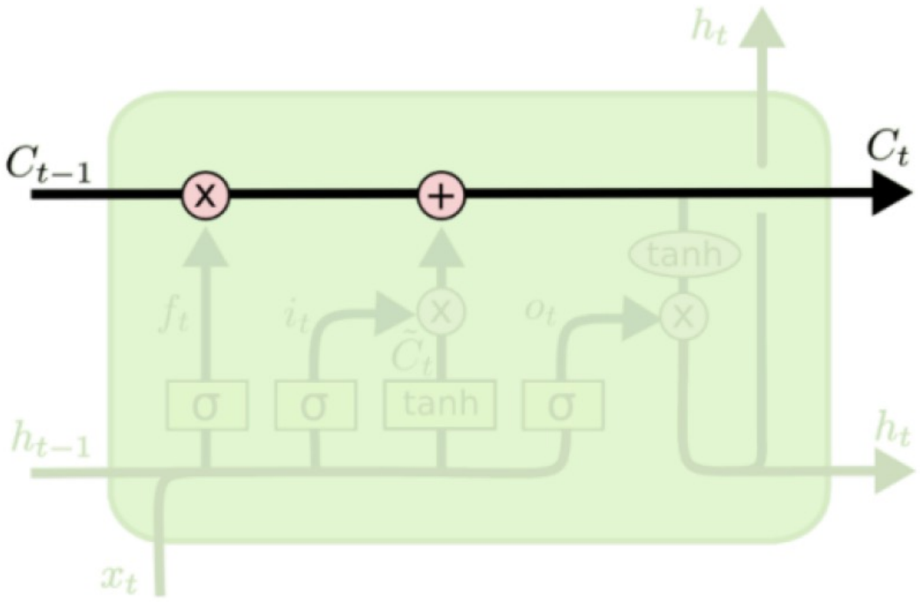
长短期记忆神经网络（Long Short-Term Memory, LSTM ）

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer there are four, interacting in a very special way.

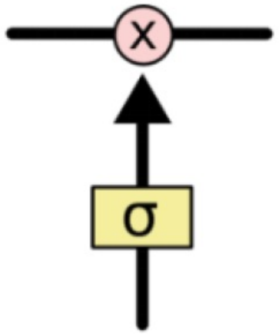


The repeating module in an LSTM contains four interacting layers.

The core idea behind LSTMs: Cell State

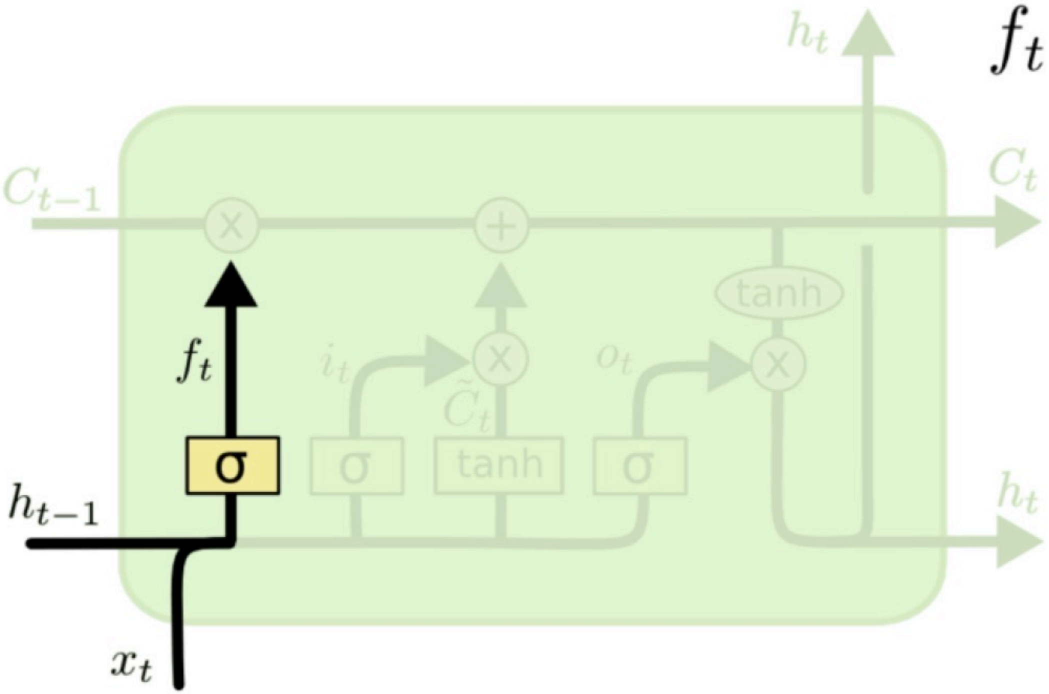


Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.



An LSTM has three of these gates, to protect and control the cell state.

LSTM: Forget gate



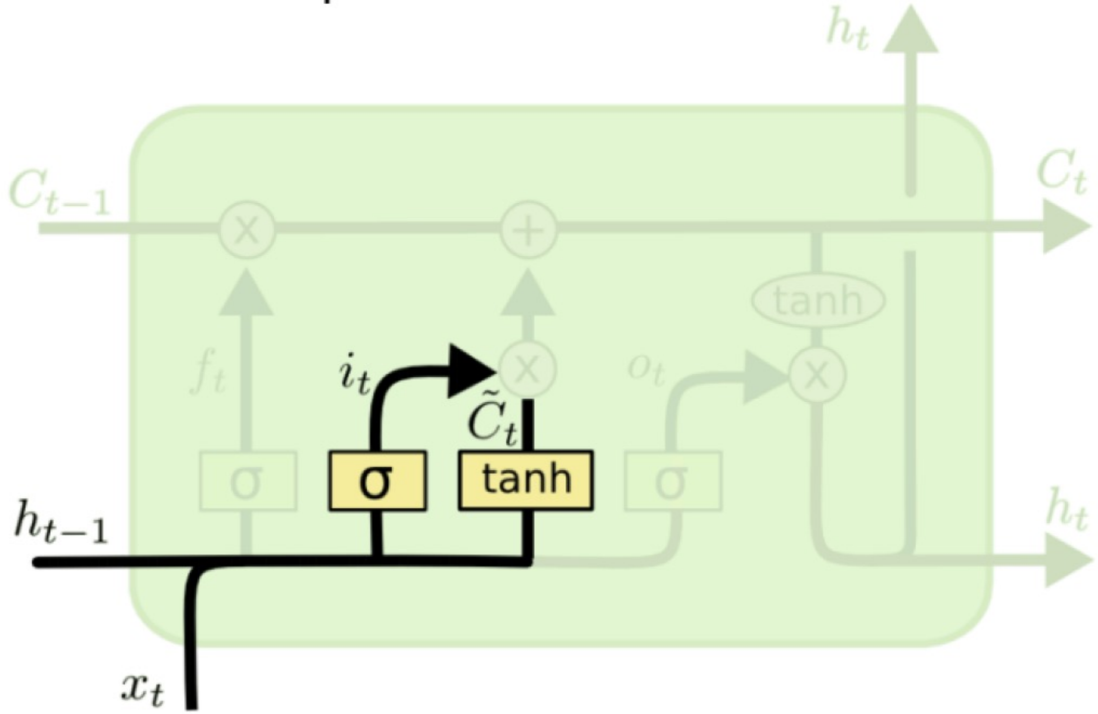
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

It looks at h_{t-1} and x_t and outputs a number between 0 and 1 for each number in the cell state C_{t-1}.

A 1 represents “completely keep this” while a 0 represents “completely get rid of this”.

LSTM: Input gate and Cell State

The next step is to decide what new information we're going to store in the cell state.



a sigmoid layer called the "**input gate layer**" decides which values we'll update.

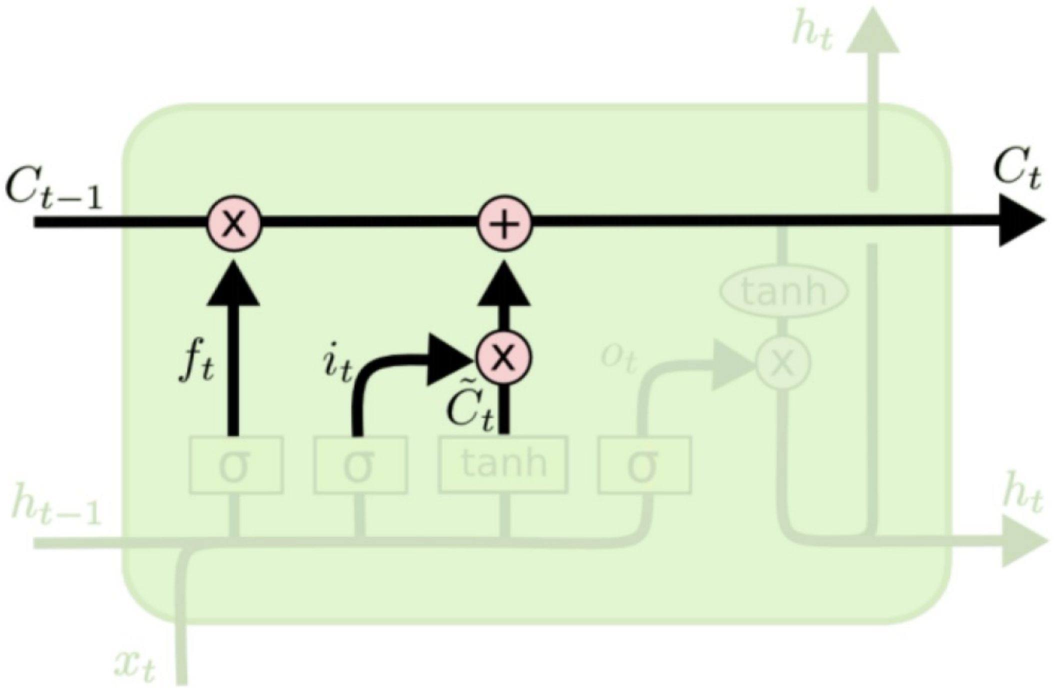
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

a tanh layer creates a vector of new candidate values, that could be added to the state.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM: Input gate and Cell State

It's now time to update the old cell state into the new cell state.



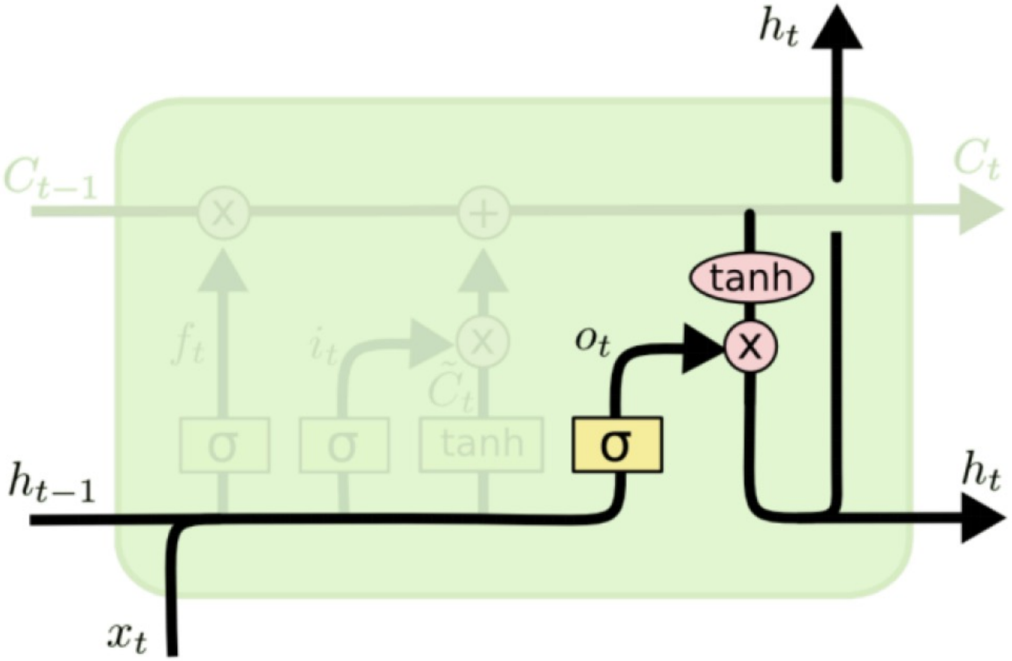
$$C_t = \boxed{f_t * C_{t-1}} + \boxed{i_t * \tilde{C}_t}$$

We multiply the old state by ft forgetting the things we decided to forget earlier.

Then, we add the new candidate values, scaled by how much we decided to update each state value.

LSTM: Output

Finally, we need to decide what we're going to output.



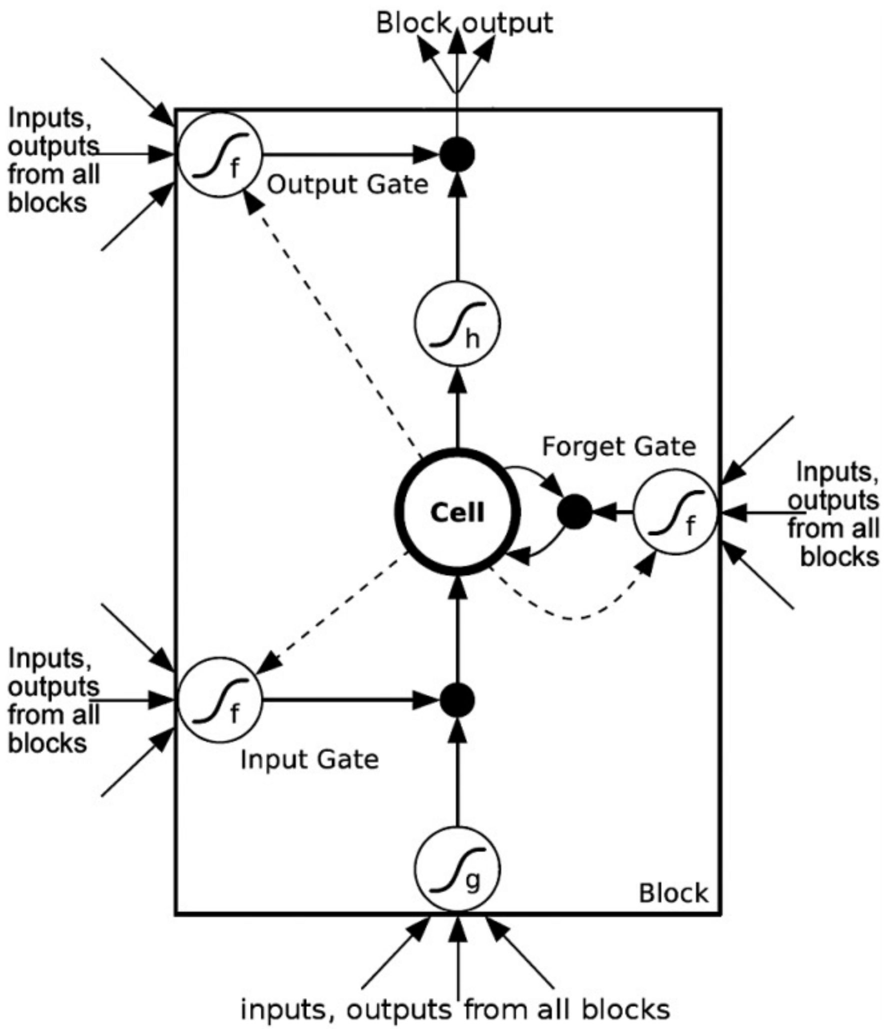
First, we run a sigmoid layer which decides what parts of the cell state we're going to output.

$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

Then, we put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

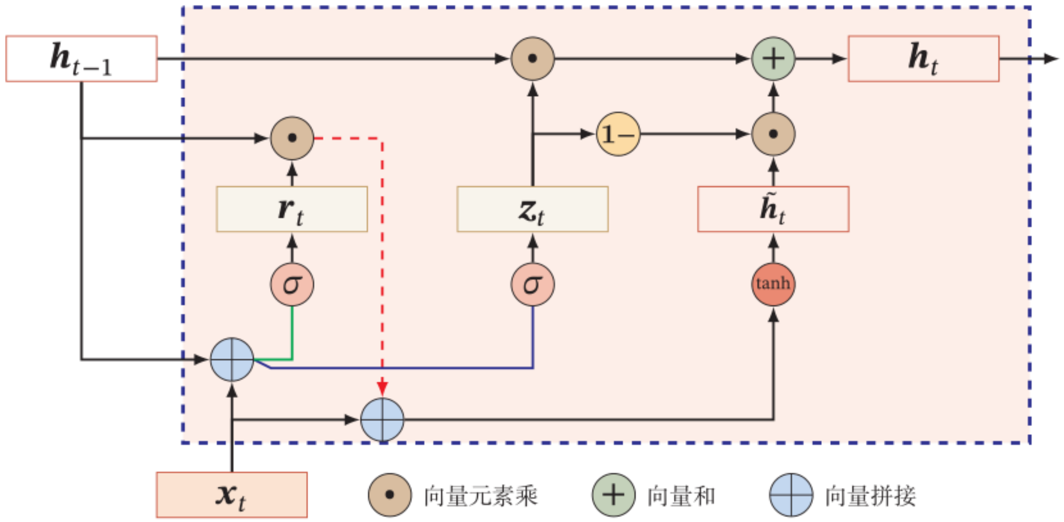
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM: How it works?



input gate	forget gate	behavior
0	1	remember the previous value
1	1	add to the previous value
0	0	erase the value
1	0	overwrite the value

Gated Recurrent Unit, GRU



重置门

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

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

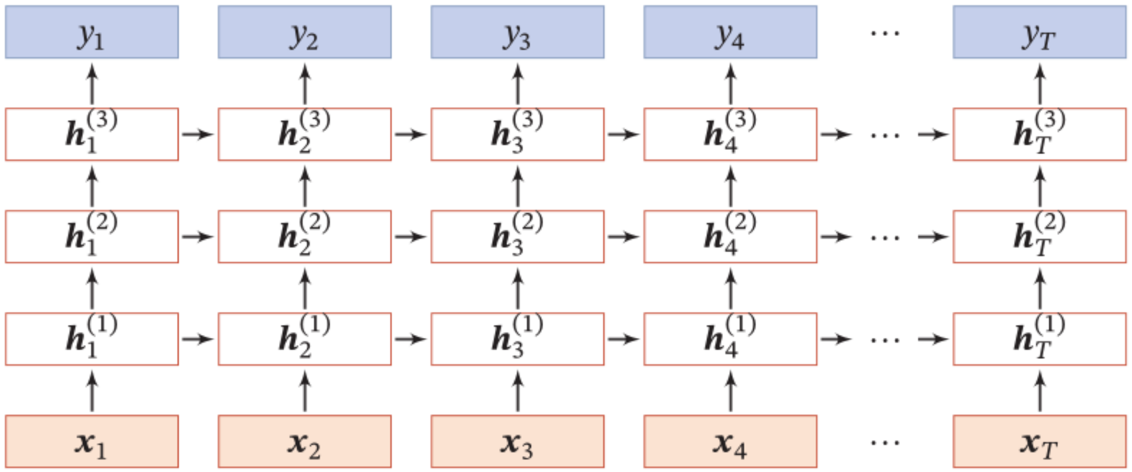
更新门

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

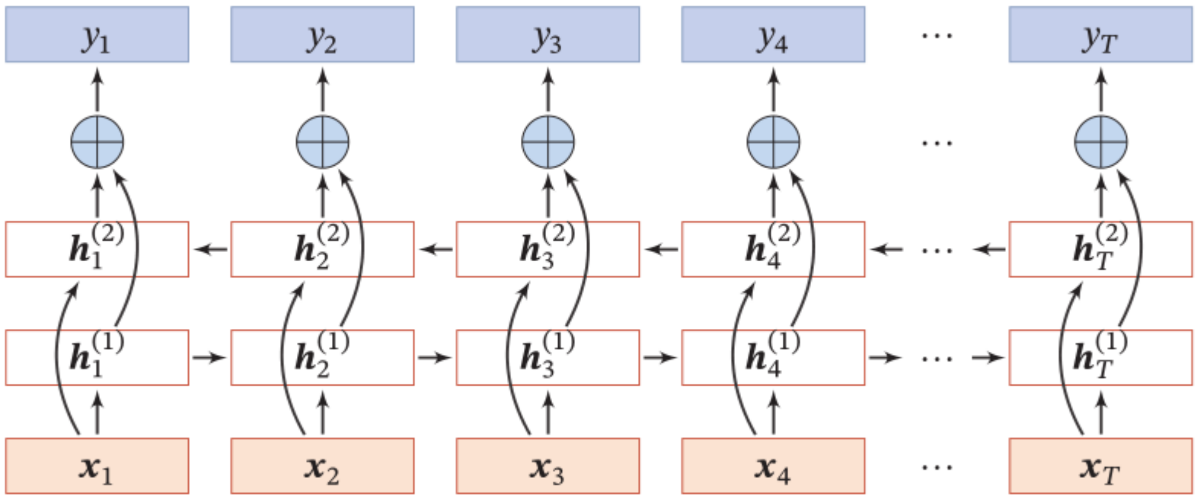
$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t,$$

深层模型

堆叠循环神经网络 (Stacked RNN)



双向循环神经网络



y_1
y_2
y_3
y_4
…
y_T
$\boldsymbol{h}_1^{(2)}$
$\boldsymbol{h}_2^{(2)}$
$\boldsymbol{h}_3^{(2)}$
$\boldsymbol{h}_4^{(2)}$
$\boldsymbol{h}_T^{(2)}$
$\boldsymbol{h}_1^{(1)}$
$\boldsymbol{h}_2^{(1)}$
$\boldsymbol{h}_3^{(1)}$
$\boldsymbol{h}_4^{(1)}$
$\boldsymbol{h}_T^{(1)}$
$\boldsymbol{x}_1$
$\boldsymbol{x}_2$
$\boldsymbol{x}_3$
$\boldsymbol{x}_4$
$\boldsymbol{x}_T$

循环网络应用

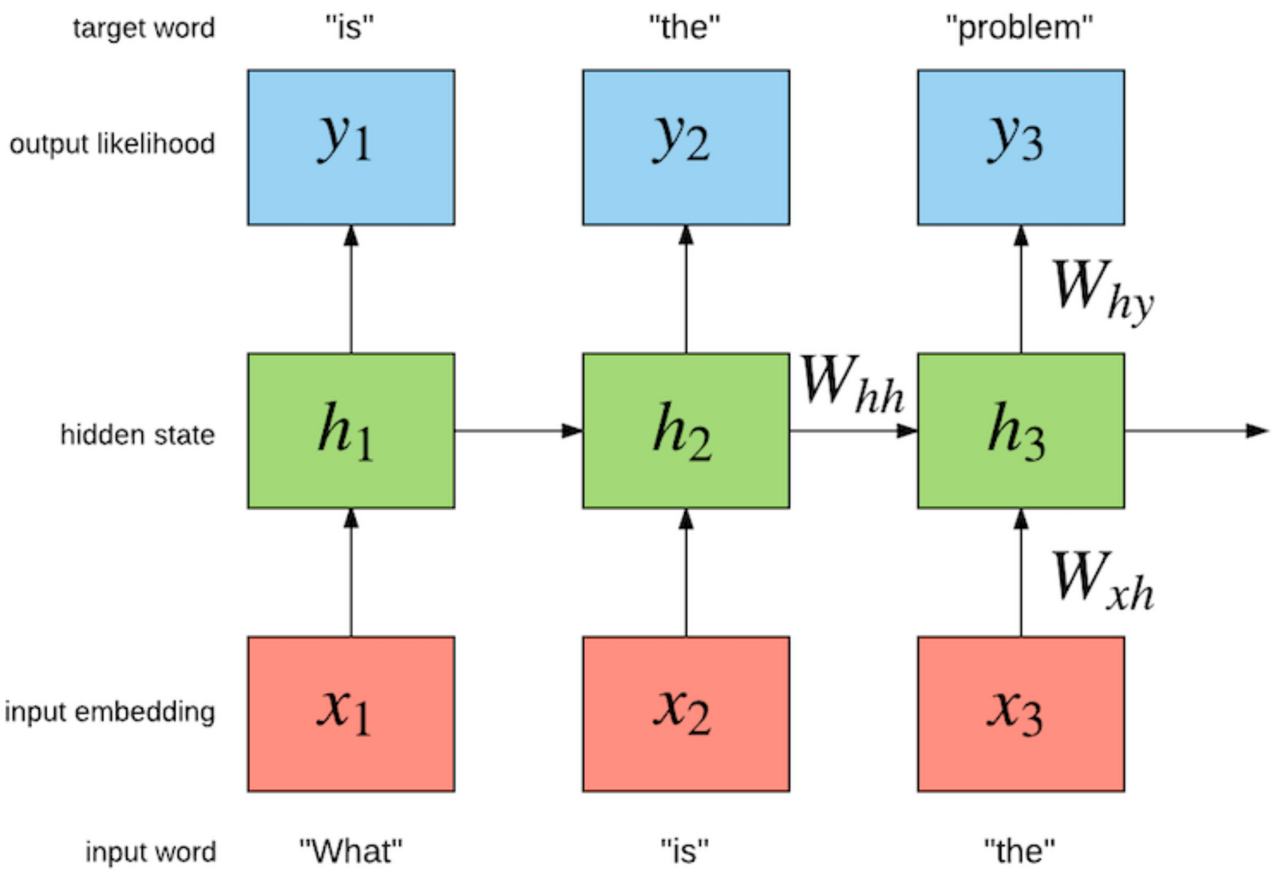
语言模型

- 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!
  - $P(x_1, x_2, \cdots, x_T)$
  - $= \prod_i P(x_i | x_{i-1}, \cdots, x_1)$
  - $\approx \prod_i P(x_i | x_{i-1}, \cdots, x_{i-n+1})$

N元语言模型

语言模型



target word
"is"
"the"
"problem"
output likelihood
y_1
y_2
y_3
W_{hy}
hidden state
h_1
h_2
W_{hh}
h_3
W_{xh}
input embedding
x_1
x_2
x_3
input word
"What"
"is"
"the"

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

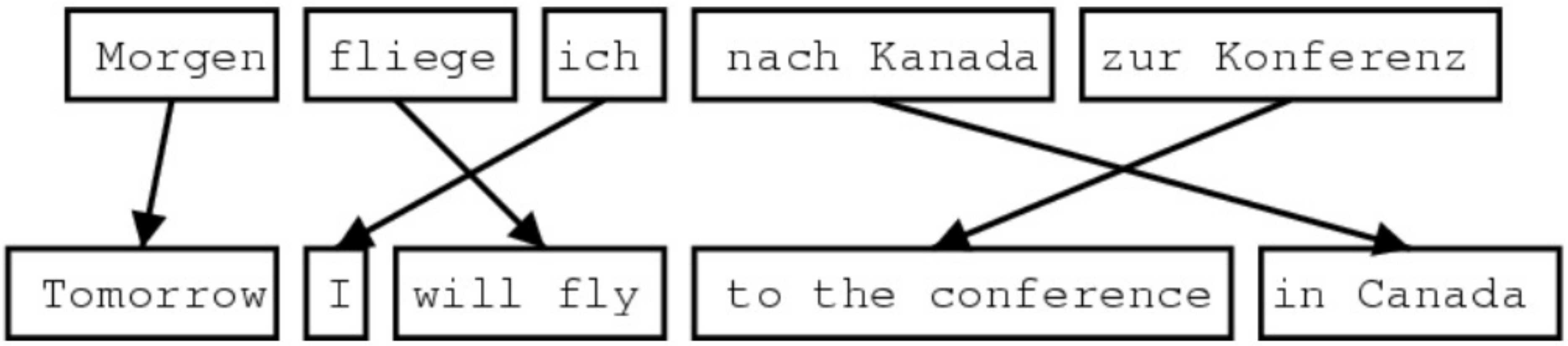
我在这里中的夜里
就像一场是一种生命的意叶
就像我的生活变得在我一样
可我们这是一个知道
我只是一天你会怎吗
可我们这是我们的是不要为你
我们想这有一种生活的时候

作诗

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.	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?

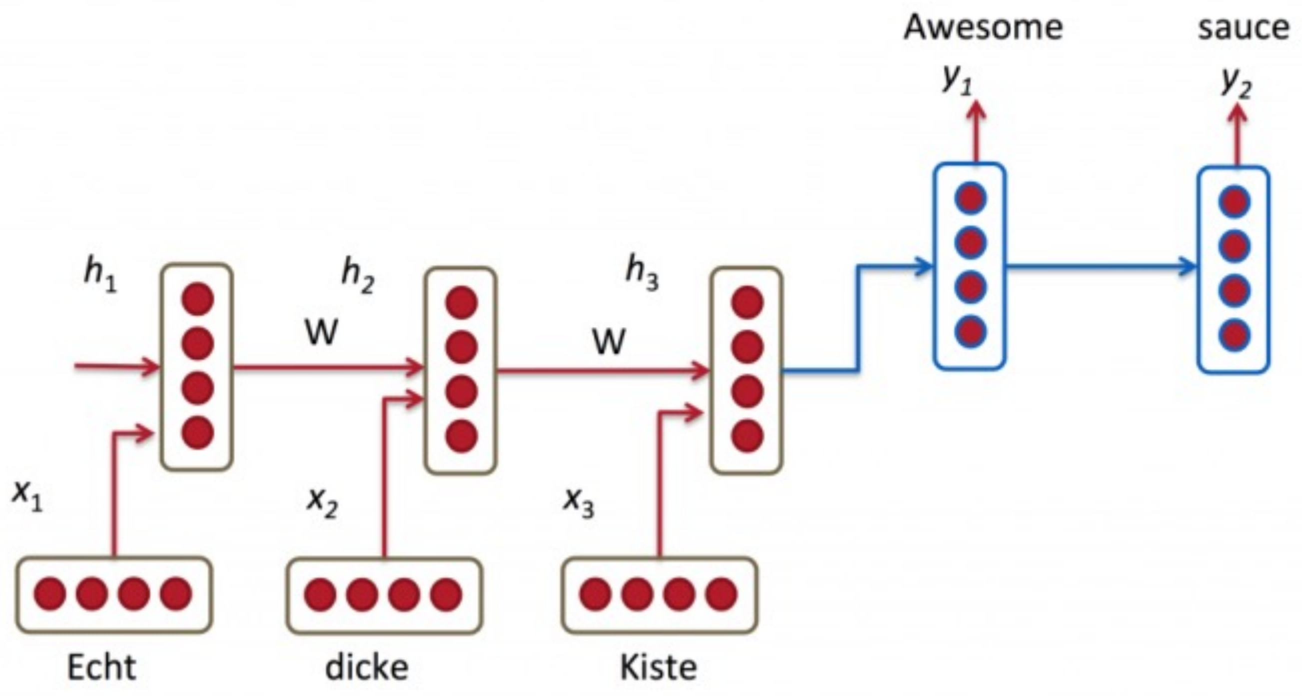
传统统计机器翻译

- 源语言：f
- 目标语言：e
  - 模型：$\hat{e} = \text{argmax}_e\, p(e|f) \;=\; \text{argmax}_e\, p(f|e)p(e)$
  - p(f|e): 翻译模型
  - p(e) : 语言模型

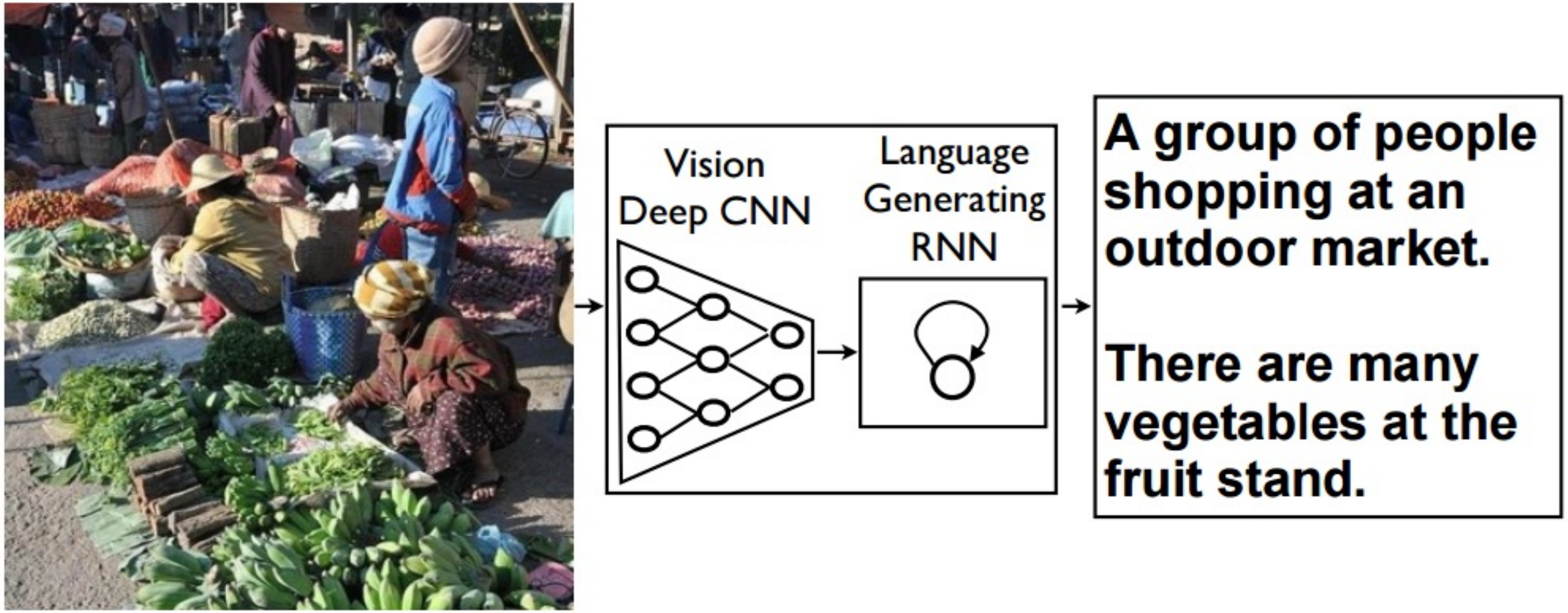


基于序列到序列的机器翻译

- 一个RNN用来编码
- 另一个RNN用来解码



看图说话


Vision
Deep CNN
Language
Generating
RNN
A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

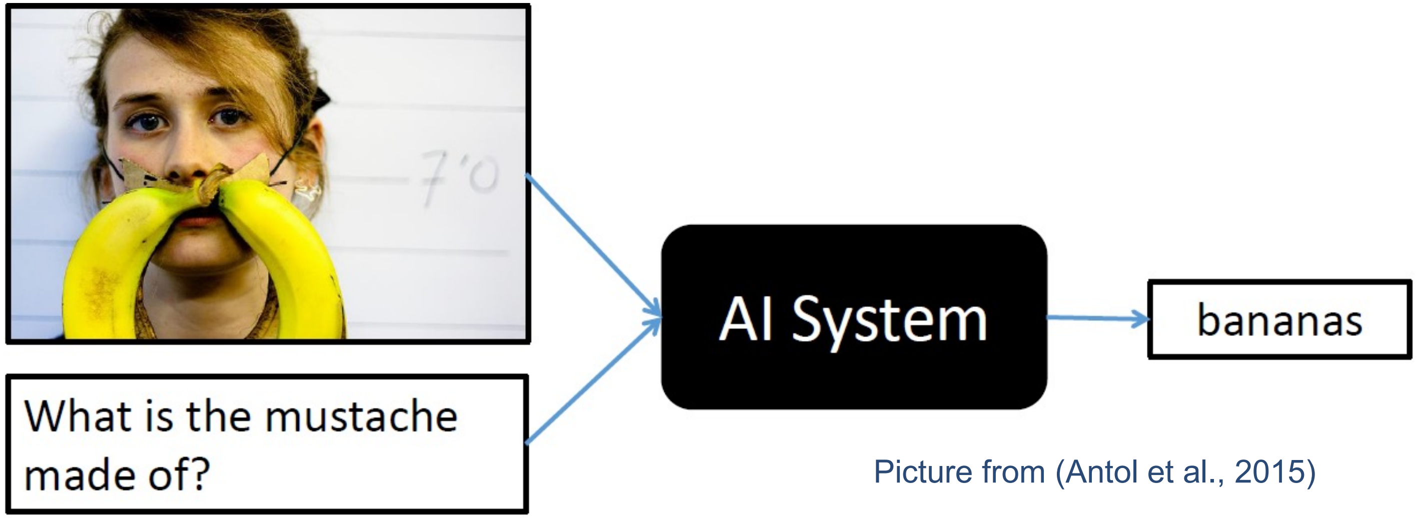
看图说话

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

Visual Question Answering (VQA)

Demo Website

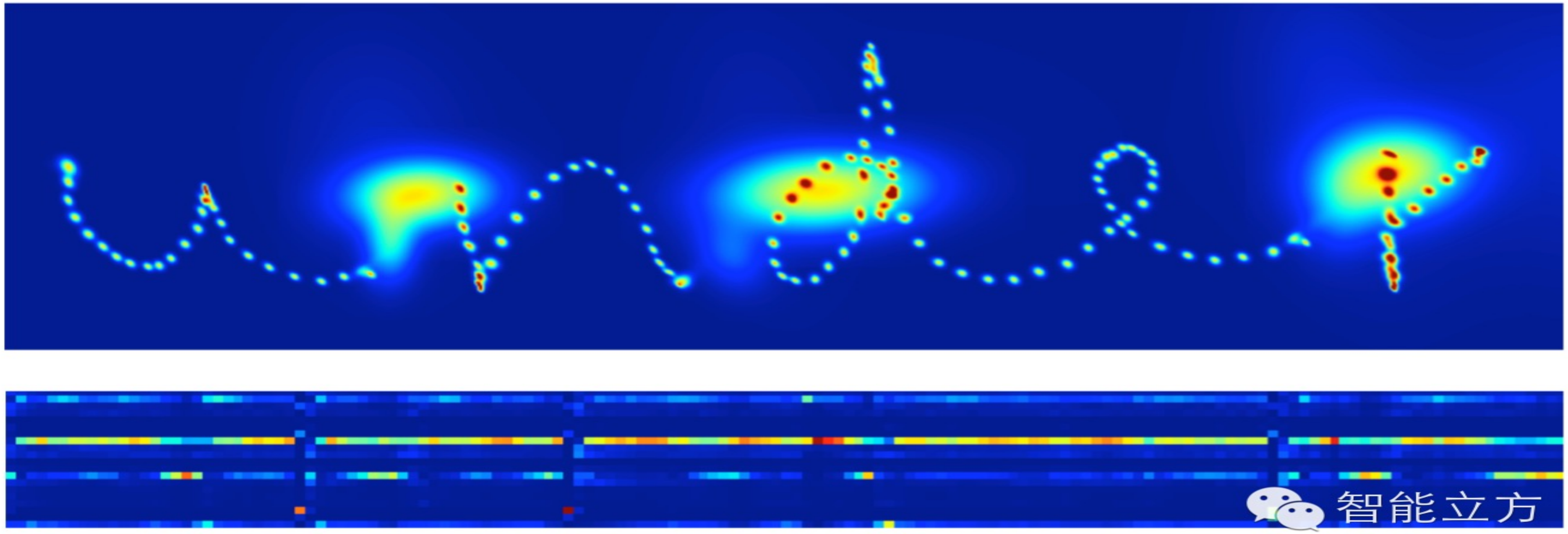
VQA: Given an image and a natural language question about the image, the task is to provide an accurate natural language answer



Picture from (Antol et al., 2015)

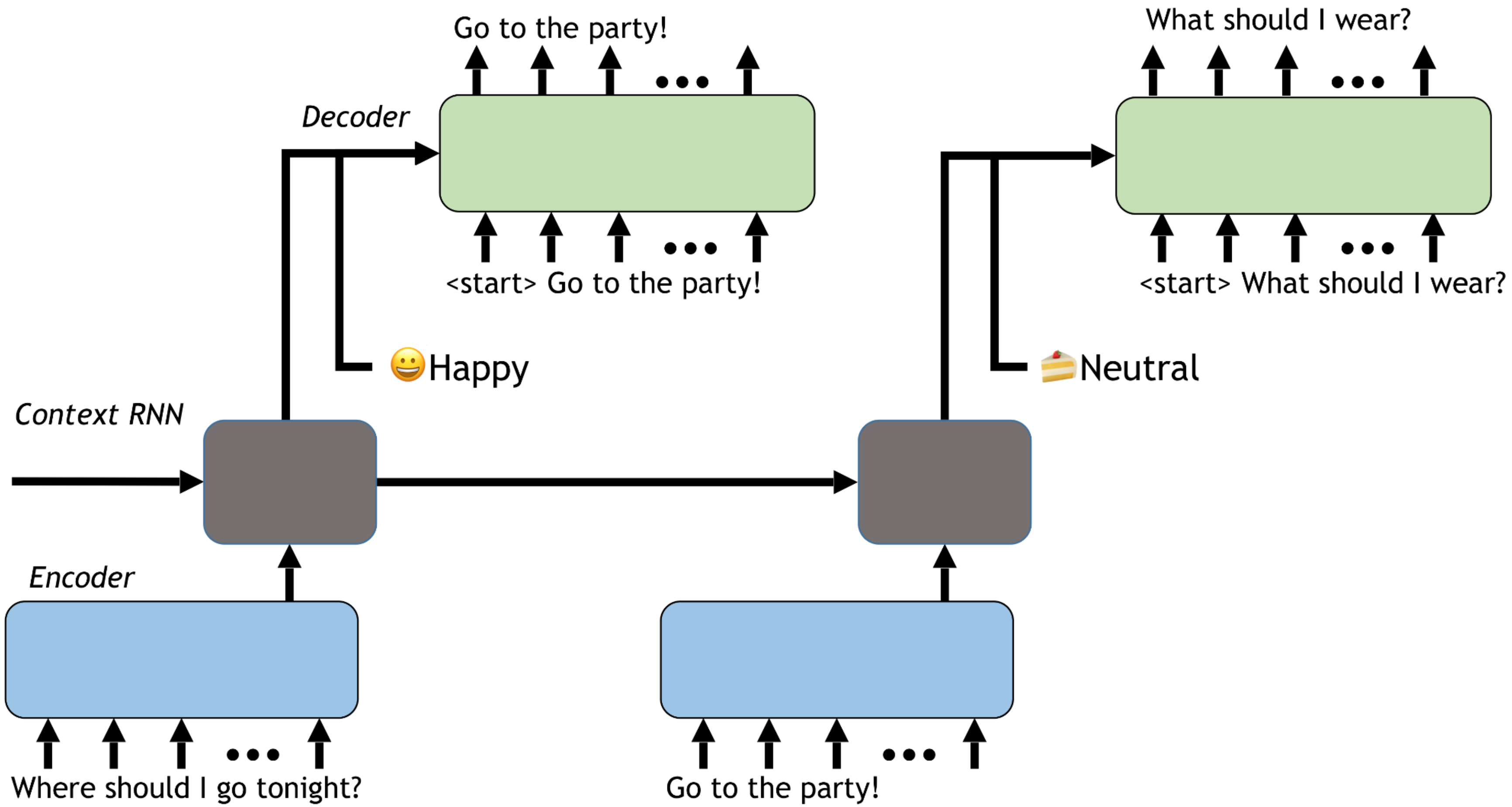
写字

- 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(offset x, 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