《神经网络与深度学习》

循环神经网络

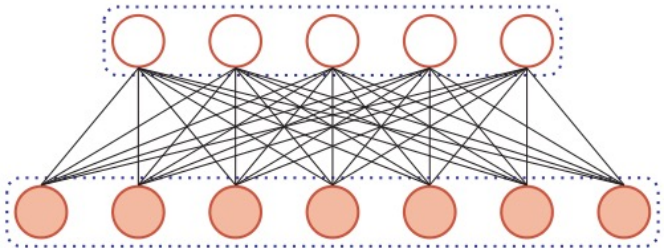
https://nndl.github.io/

内容

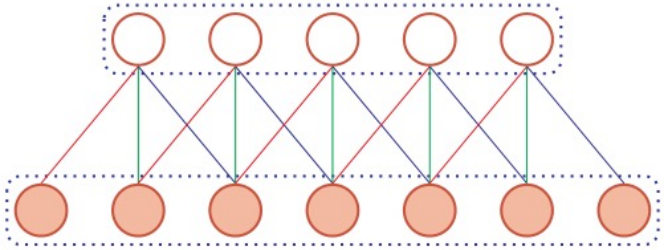
- 循环神经网络
  - 简单循环神经网络
  - 应用到机器学习
  - 基于门控的循环神经网络
- 循环神经网络的扩展
  - 典型的循环神经网络(Stacked LSTM, Bi-LSTM, Lattice-LSTM, Grid LSTM…)+前馈网络(Highway Network)
  - 扩展到图结构(RecursiveNN, GNN, GCN…)
  - 循环神经网络的应用

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。



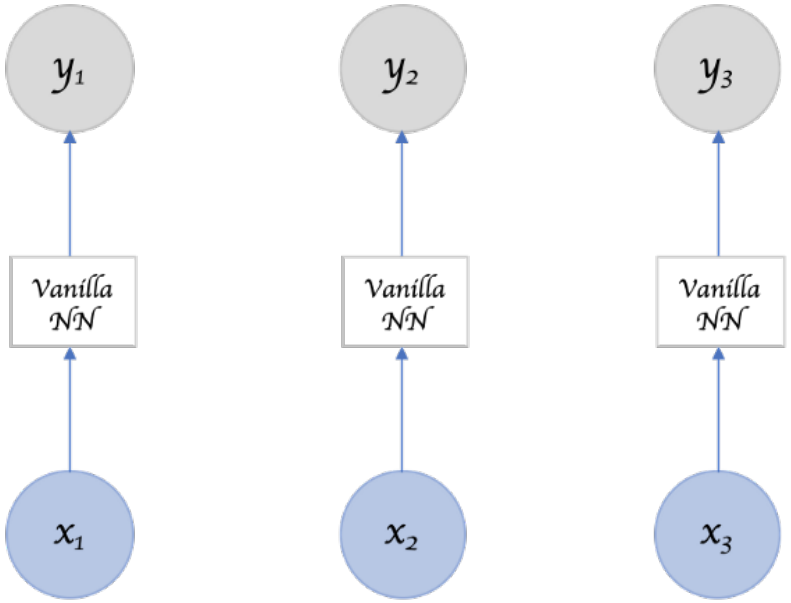
(a) 全连接层



(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。



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

$$y_t = w_0 + \sum_{k=1}^{K} w_k y_{t-k} + \epsilon_t$$

  - $\epsilon_t \sim N(0, \sigma^2)$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）

$$y_t = f(x_t, x_{t-1}, \cdots, x_{t-K_x}, y_{t-1}, y_{t-2}, \cdots, y_{t-K_y})$$

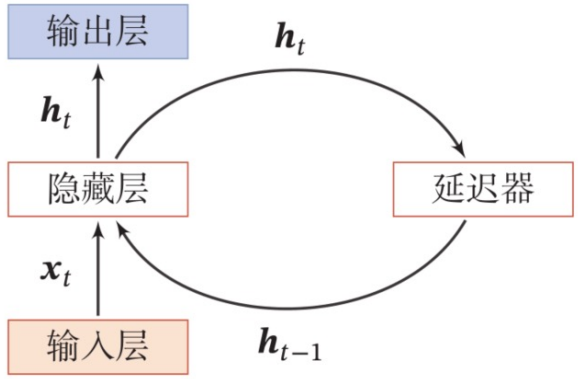
  - 其中 $f(\cdot)$ 表示非线性函数，可以是一个前馈网络，K_x 和 K_y 为超参数.

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。

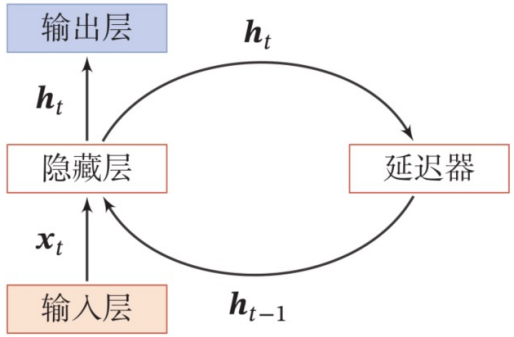
$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

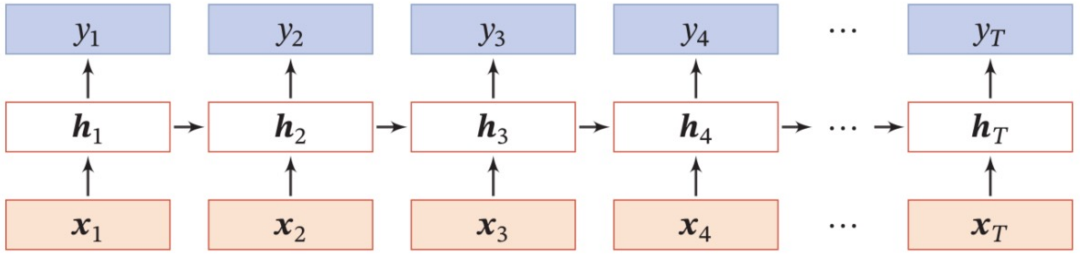
活性值
状态(state/hidden state)



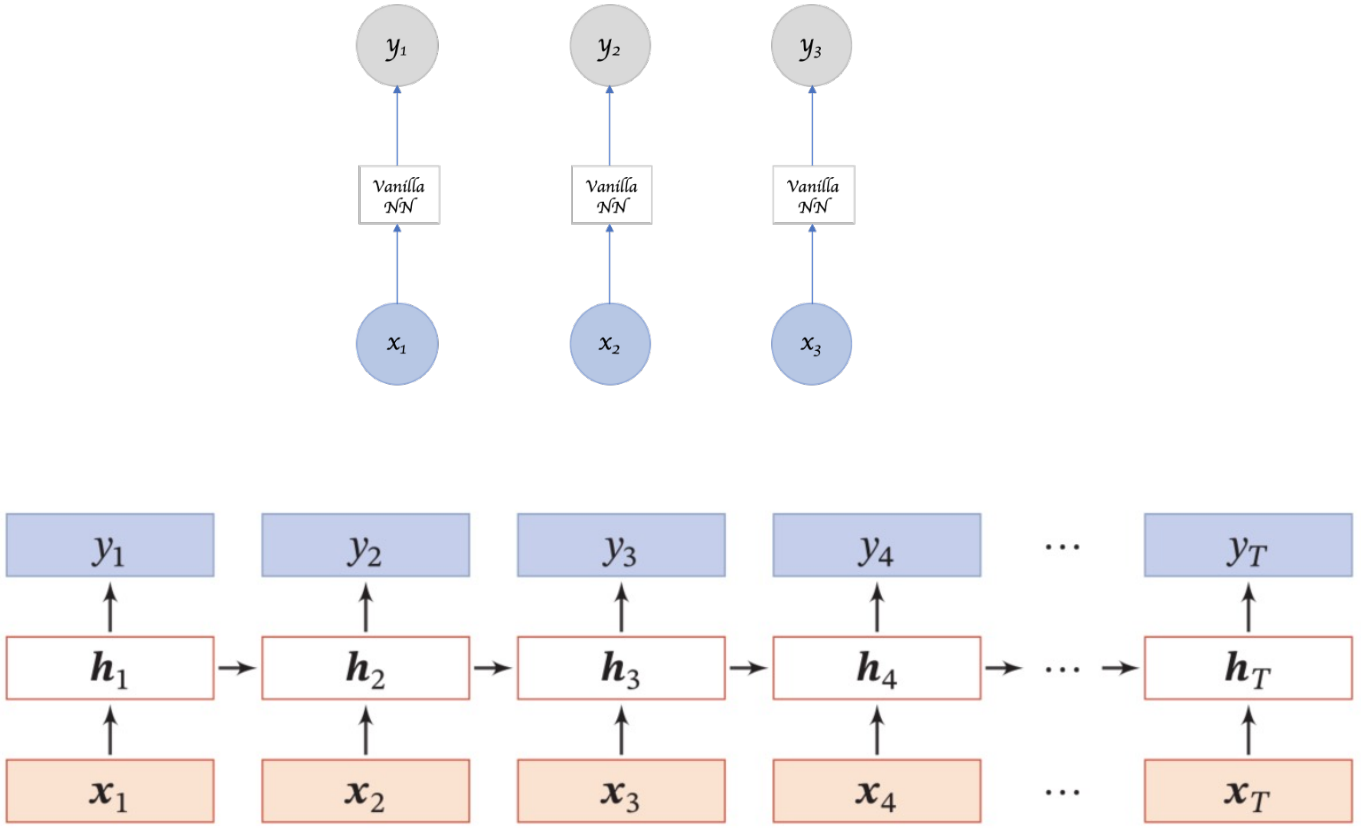
- 循环神经网络比前馈神经网络更加符合生物神经网络的结构。
- 循环神经网络已经被广泛应用在语音识别、语言模型以及自然语言生成等任务上

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

FFN v.s. RNN


y_1
y_2
y_3
Vanilla NN
Vanilla NN
Vanilla NN
x_1
x_2
x_3
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 ）

- 只有一个隐藏层，隐藏层的状态在t时刻的更新公式为：

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

其中$U \in R^{D\times D}$为状态-状态权重矩阵， $W \in R^{D\times M}$为状态-输入权重矩阵，$b \in R^{D}$为偏置向量，$f(.)$为非线性激活函数，通常为Logistic函数或Tanh函数。

- 网络在t时刻的输出可用一个全连接网络层实现：

$$\boldsymbol{y}_t = \boldsymbol{V}\boldsymbol{h}_t,$$

其中$V \in R^{M'\times D}$

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

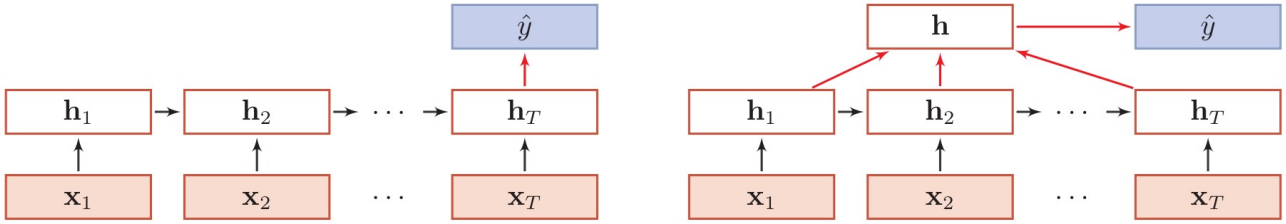
应用到机器学习

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

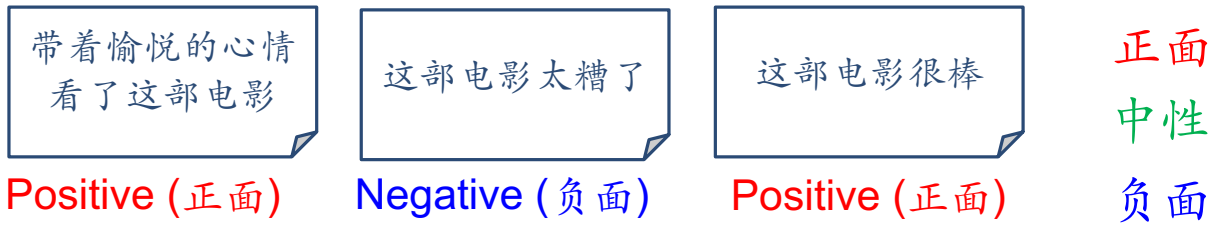
- 序列到类别

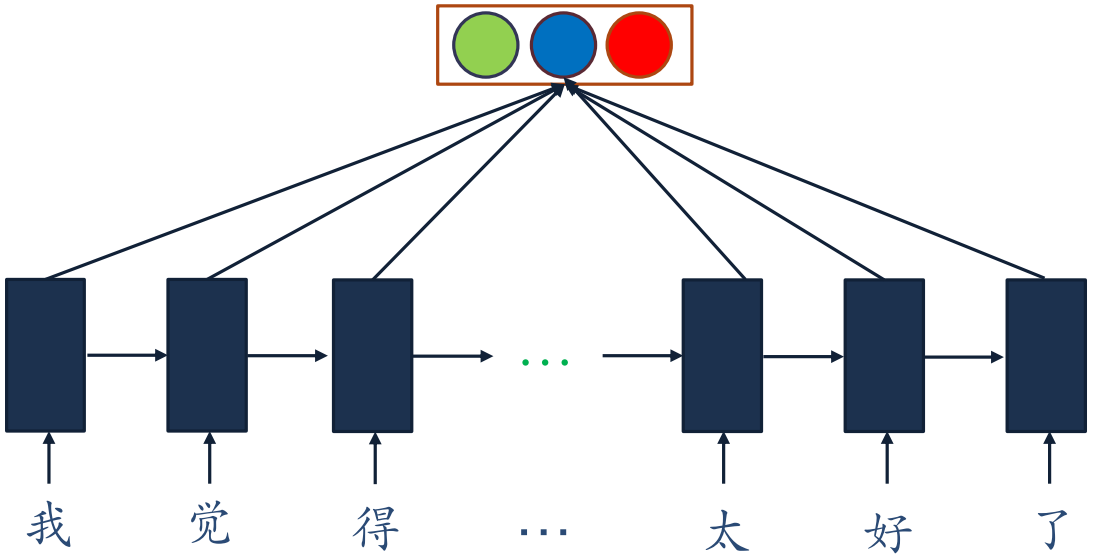


(a) 正常模式　　(b) 按时间进行平均采样模式

序列到类别

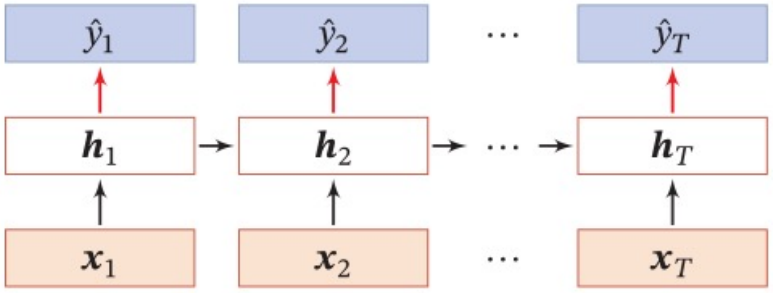
- 情感分类


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)
正面
中性
负面


我
觉
得
…
太
好
了

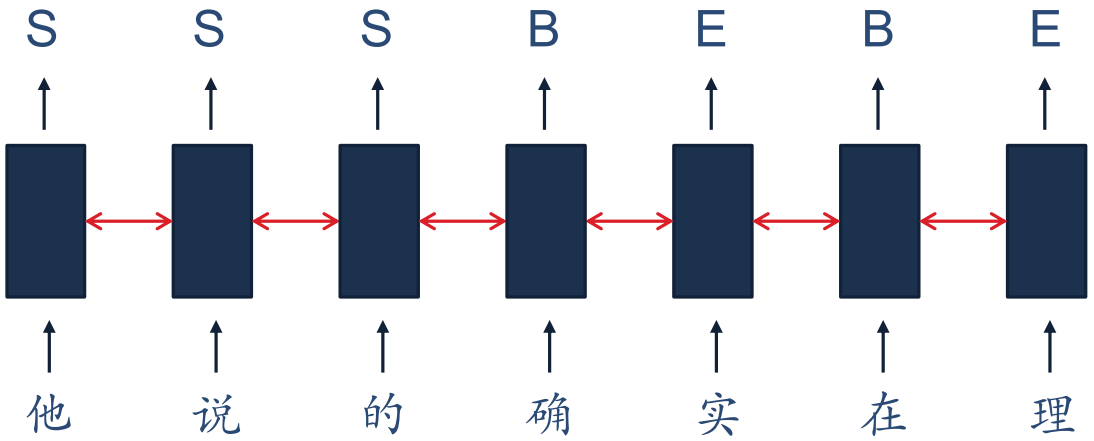
应用到机器学习

- 同步的序列到序列模式



同步的序列到序列模式

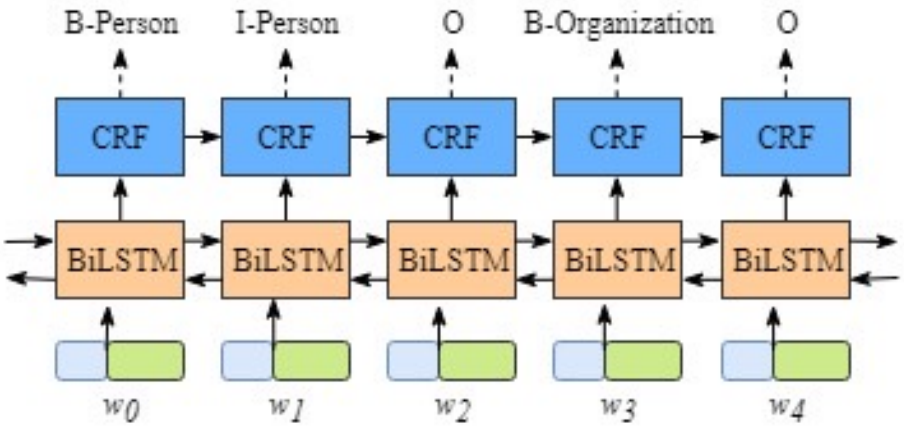
- 中文分词



同步的序列到序列模式

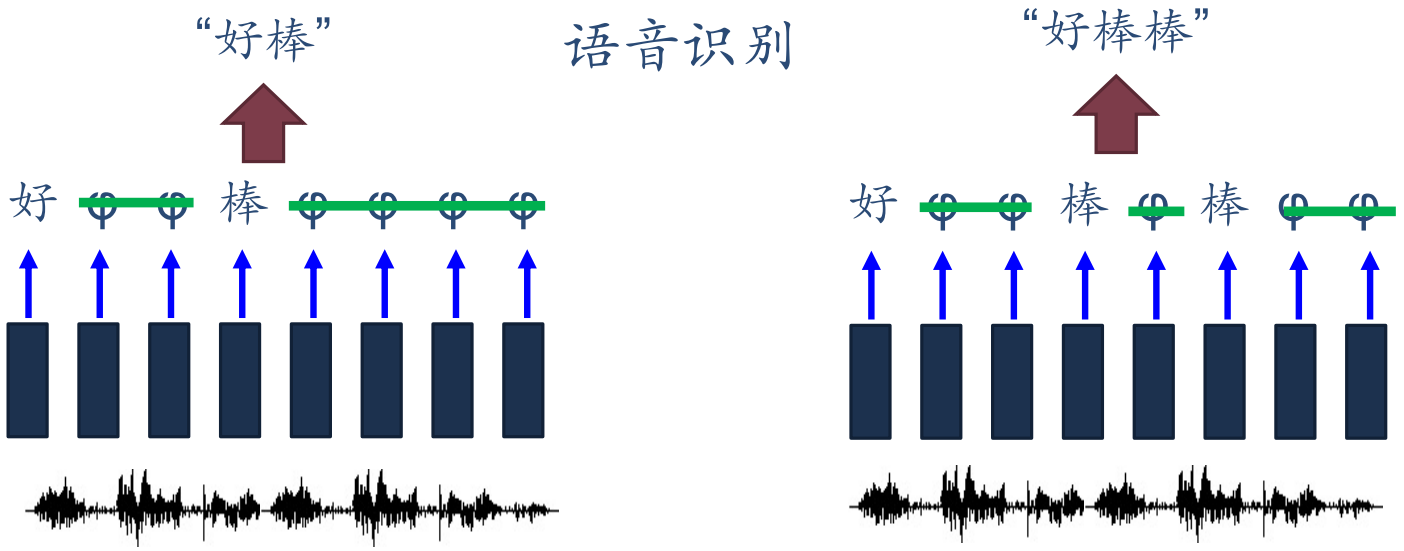
- 信息抽取(Information Extraction，IE)
  - 从无结构的文本中抽取结构化的信息，形成知识

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%。



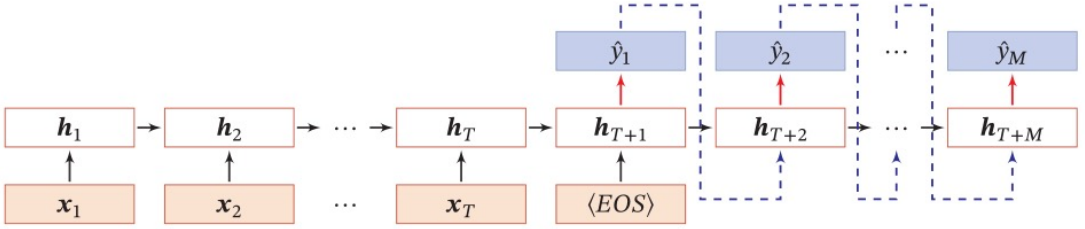
同步的序列到序列模式

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



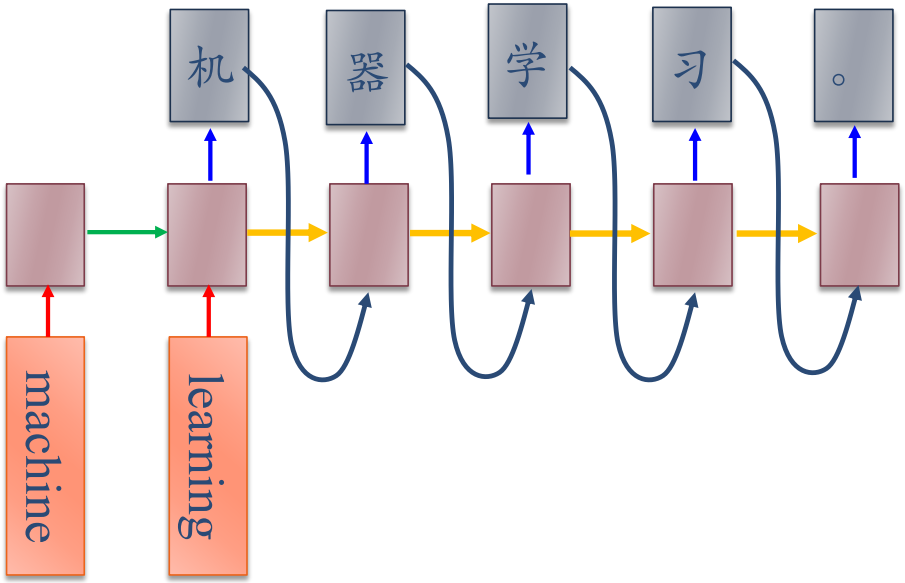
应用到机器学习

- 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(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

$$\mathcal{L}_t = \mathcal{L}(\mathbf{y}_t, g(\mathbf{h}_t)),$$

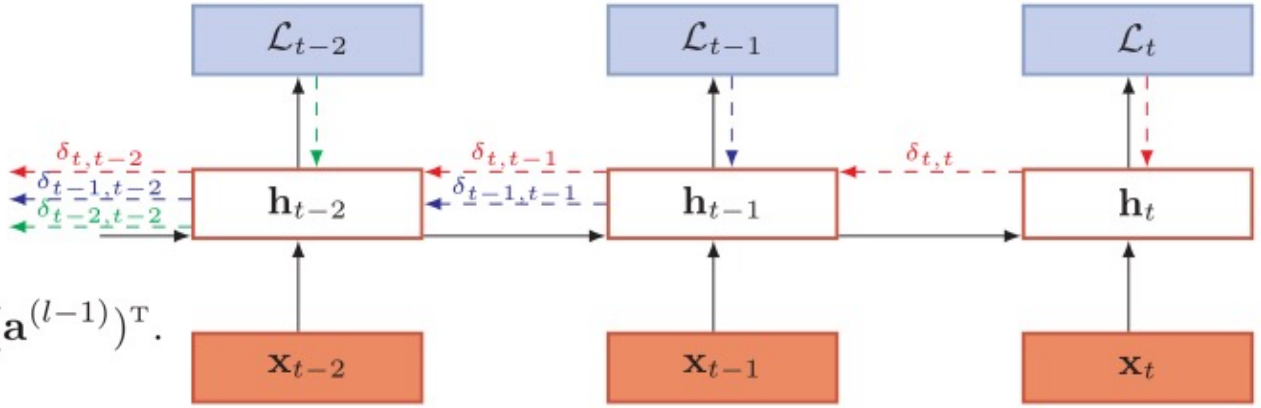
- 总损失函数

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

梯度

$$h_t = f(z_t) = f(Uh_{t-1} + Wx_t + b)$$

- 随时间反向传播算法



$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^{\mathrm{T}}.$$

$$\delta^{(l)} \triangleq \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} = \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l+1)}}$$

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

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

$\delta_{t,k}$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

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

若$\gamma > 1$，当$t-k \to \infty$时，$\gamma^{t-k} \to \infty$. 当间隔$t-k$比较大时，梯度也变得很大，会造成系统不稳定，称为梯度爆炸问题（Gradient Exploding Problem）.

相反，若$\gamma < 1$，当$t-k \to \infty$时，$\gamma^{t-k} \to 0$. 当间隔$t-k$比较大时，梯度也变得非常小，会出现和深层前馈神经网络类似的梯度消失问题（Vanishing Gradient Problem）.

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

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

$$\frac{\partial h_t}{\partial h_{t-1}} = I$$

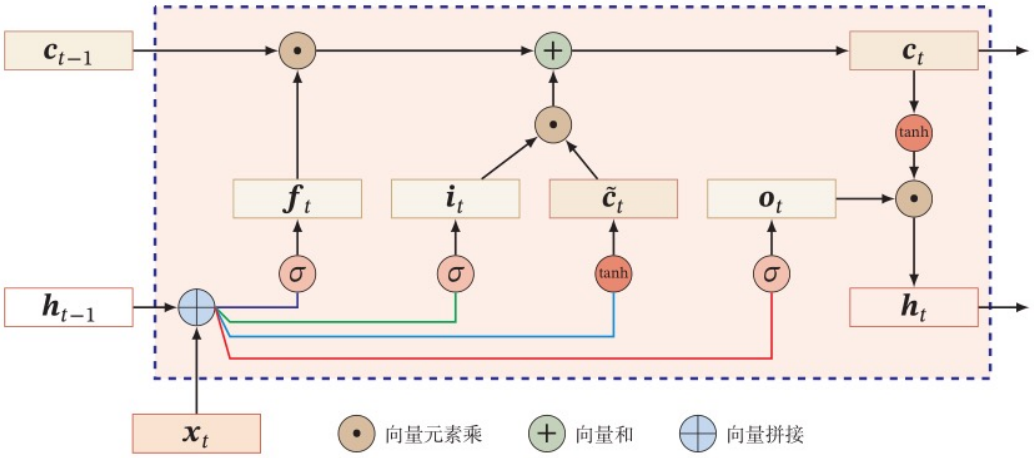
失去了非线性激活的性质

  - 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 ）



$f_t = 0, i_t = 1$
$f_t = 1, i_t = 0$

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

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

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

候选状态 $\tilde{\mathbf{c}}_t = \tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c)$

内部状态 $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,$

$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$

LSTM的各种变体

- 没有遗忘门

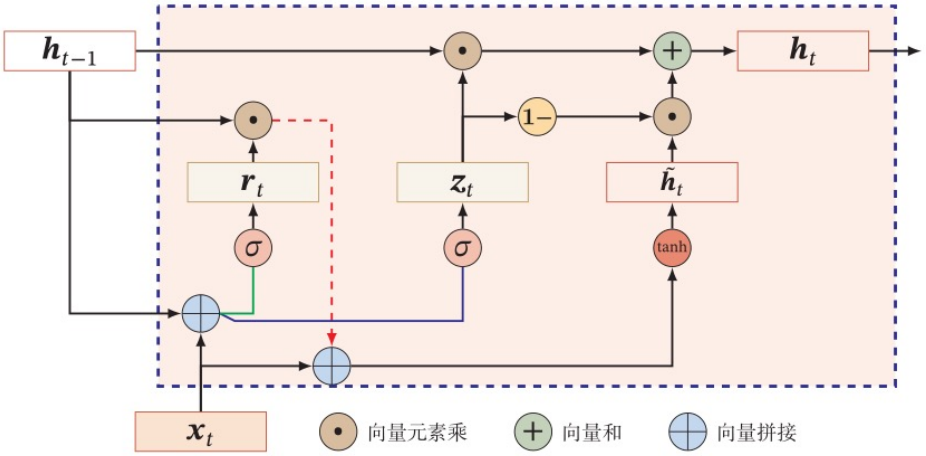
$$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t.$$

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

- peephole连接

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

Gated Recurrent Unit, GRU



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

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

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

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

内容

- 循环神经网络
  - 简单循环神经网络
  - 应用到机器学习
  - 基于门控的循环神经网络
- 循环神经网络的扩展
  - 典型的循环神经网络(Stacked LSTM, Bi-LSTM, Grid LSTM, Lattice-LSTM …)+前馈网络(Highway Network)
  - 扩展到图结构(RecNN, GCN…)
  - 循环神经网络的应用

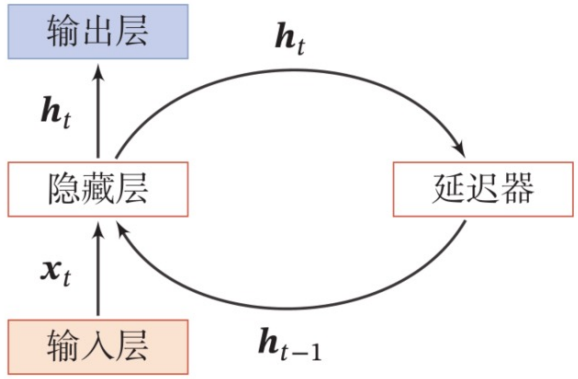
典型的循环神经网络

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

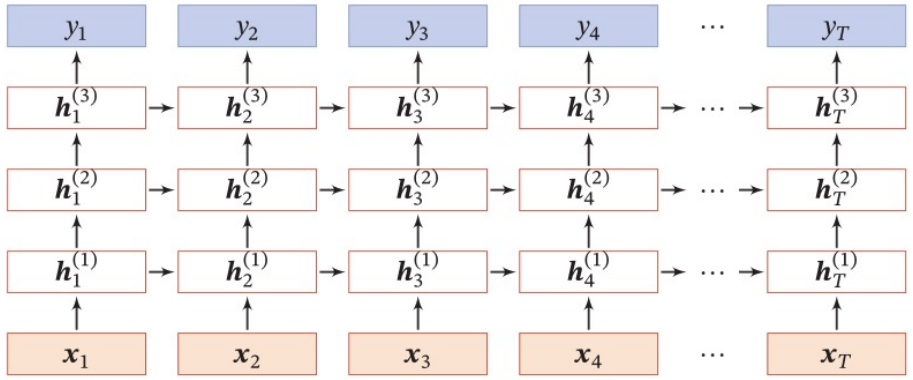
- 循环神经网络通过使用带自反馈的神经元，能够处理任意长度的时序数据。

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

活性值
状态(state/hidden state)



堆叠循环神经网络（Stacked RNN）

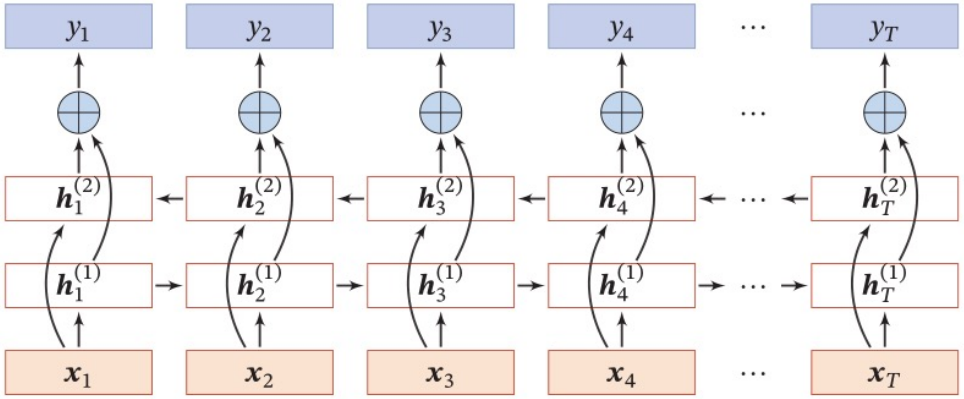


$$\boldsymbol{h}_t^{(l)} = f(\boldsymbol{U}^{(l)}\boldsymbol{h}_{t-1}^{(l)} + \boldsymbol{W}^{(l)}\boldsymbol{h}_t^{(l-1)} + \boldsymbol{b}^{(l)}),$$

Stacked LSTM的简单实现

```
class Text_Stacked_LSTM(nn.Module):
    def __init__(self):
        super(Text_Stacked_LSTM, self).__init__()

        # 指定 num_layers 参数
        self.lstm = nn.LSTM(input_size=n_class, hidden_size=n_hidden, num_layers=n_layers)
        self.W = nn.Parameter(torch.randn([n_hidden, n_class]).type(data_type))
        self.b = nn.Parameter(torch.randn([n_class]).type(data_type))

    def forward(self, x):
        batch_size = len(x)
        x = x.transpose(0, 1)

        # hidden_state 和 cell_state 的维度是相同的  num_layers * num_directions = 2
        init_hidden_state = Variable(torch.zeros(n_layers * 1, batch_size, n_hidden))
        init_cell_state = Variable(torch.zeros(n_layers * 1, batch_size, n_hidden))

        # 这里使用最后的 output 结果进行分类预测 所以没有改变，如果要用到最后的hidden_state 和 cell_state 的话那么需要对应改变
        outputs, (_, _) = self.lstm(x, (init_hidden_state, init_cell_state))
        outputs = outputs[-1]
        model = torch.mm(outputs, self.W) + self.b
```

双向循环神经网络(Bi-directional RNN)

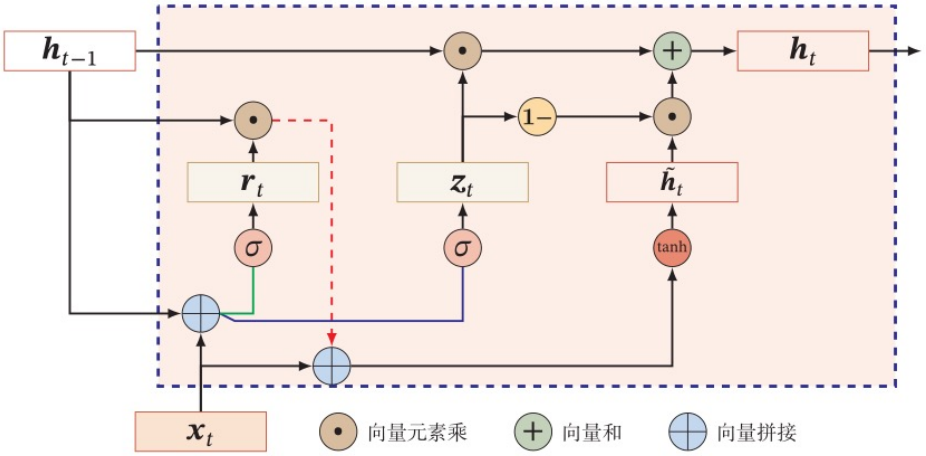


$$\boldsymbol{h}_t^{(1)} = f(\boldsymbol{U}^{(1)}\boldsymbol{h}_{t-1}^{(1)} + \boldsymbol{W}^{(1)}\boldsymbol{x}_t + \boldsymbol{b}^{(1)}),$$
$$\boldsymbol{h}_t^{(2)} = f(\boldsymbol{U}^{(2)}\boldsymbol{h}_{t+1}^{(2)} + \boldsymbol{W}^{(2)}\boldsymbol{x}_t + \boldsymbol{b}^{(2)}),$$
$$\boldsymbol{h}_t = \boldsymbol{h}_t^{(1)} \oplus \boldsymbol{h}_t^{(2)},$$

Bi-LSTM的简单实现

```
class Text_Bi_LSTM(nn.Module):
    def __init__(self):
        super(Text_Bi_LSTM, self).__init__()

        # 指定 bidirectional = True
        self.lstm = nn.LSTM(input_size=n_class, hidden_size=n_hidden, bidirectional=True)
        self.W = nn.Parameter(torch.randn([2*n_hidden, n_class]).type(data_type))
        # 注意这里 全连接层 的 W 是 num_directions * hidden_size
        self.b = nn.Parameter(torch.randn([n_class]).type(data_type))

    def forward(self, x):
        batch_size = len(x)
        x = x.transpose(0, 1)

        # hidden_state 和 cell_state 的维度是相同的  num_layers * num_directions = 2
        init_hidden_state = Variable(torch.zeros(1*2, batch_size, n_hidden))
        init_cell_state = Variable(torch.zeros(1*2, batch_size, n_hidden))

        outputs, (_, _) = self.lstm(x, (init_hidden_state, init_cell_state))
        outputs = outputs[-1]
        final_output = torch.mm(outputs, self.W) + self.b

        return final_output
```

A GRU Recap



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

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

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

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

From GRU to Highway Network

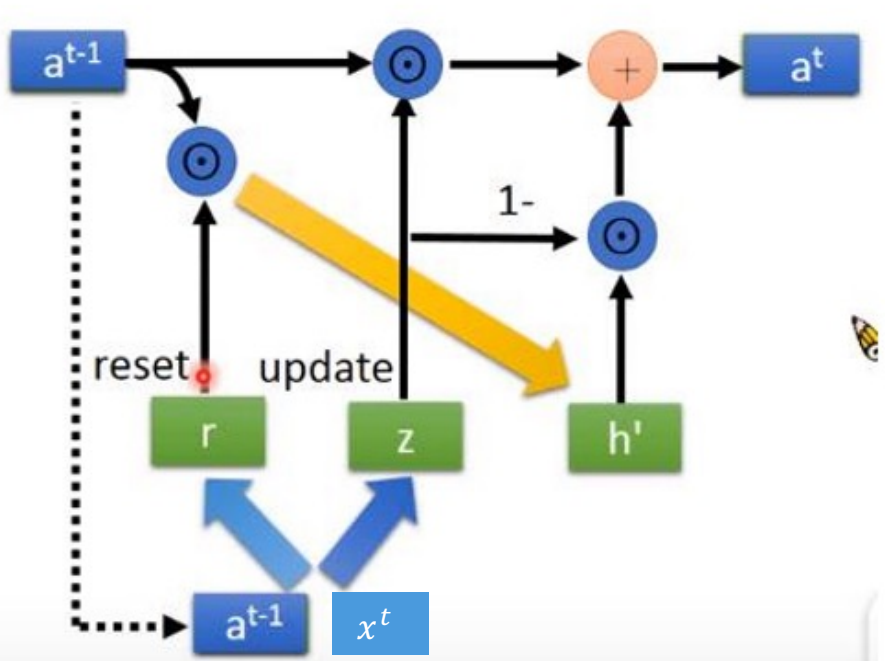
No input x^t at each step

No output y^t at each step

a^{t-1} is the output of the (t-1)-th layer

a^t is the output of the t-th layer

No reset gate



From GRU to Highway Network

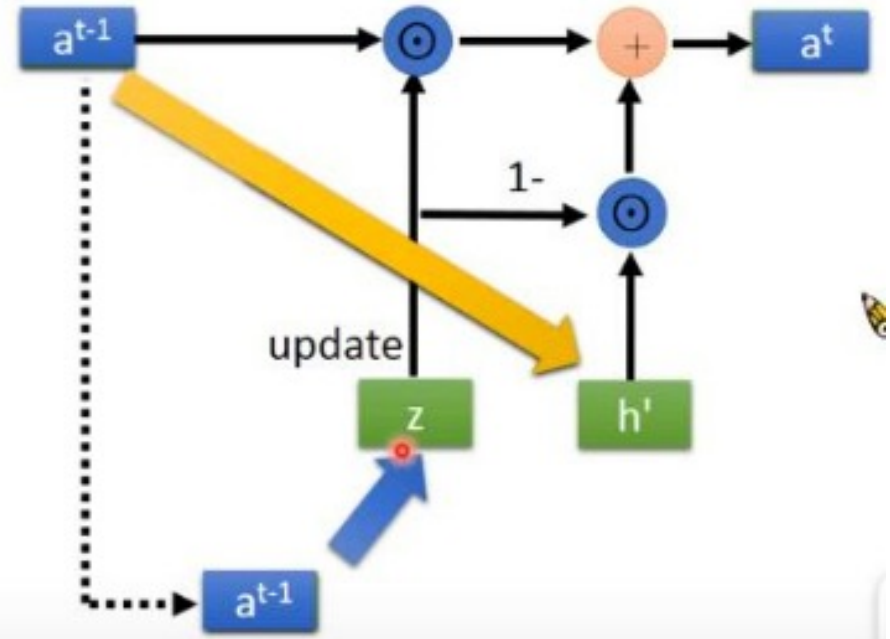
No input x^t at each step

No output y^t at each step

a^{t-1} is the output of the (t-1)-th layer

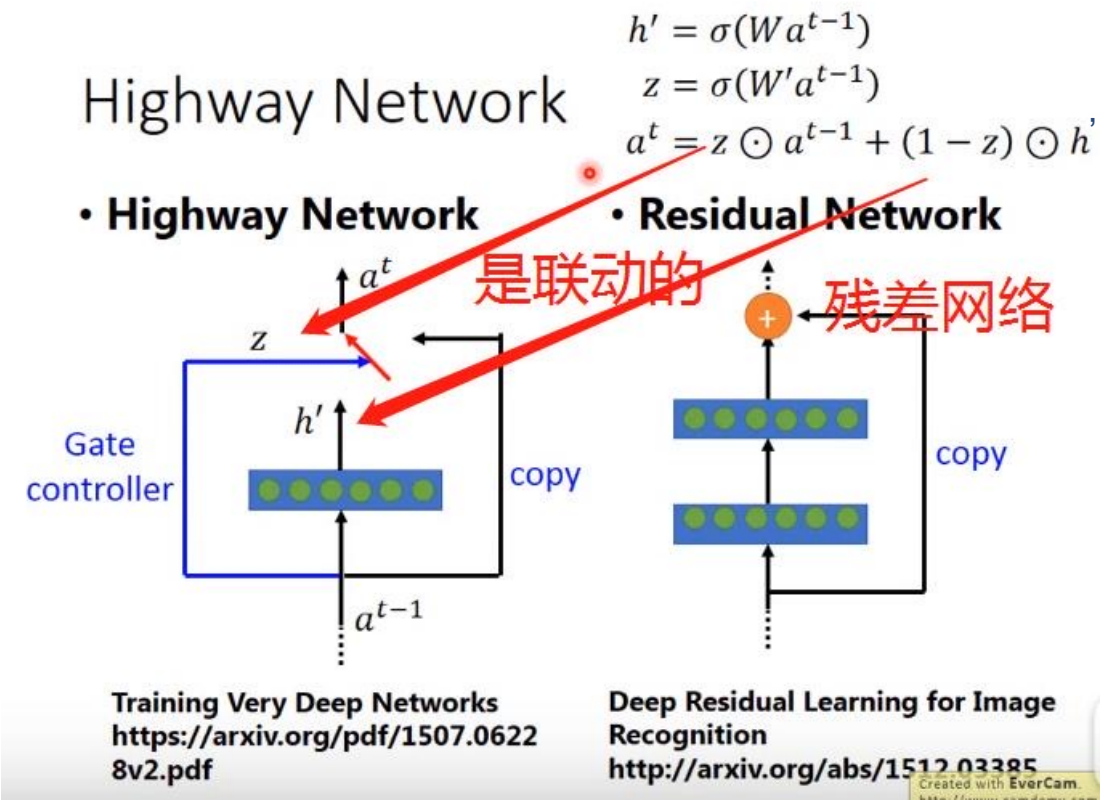
a^t is the output of the t-th layer

No reset gate

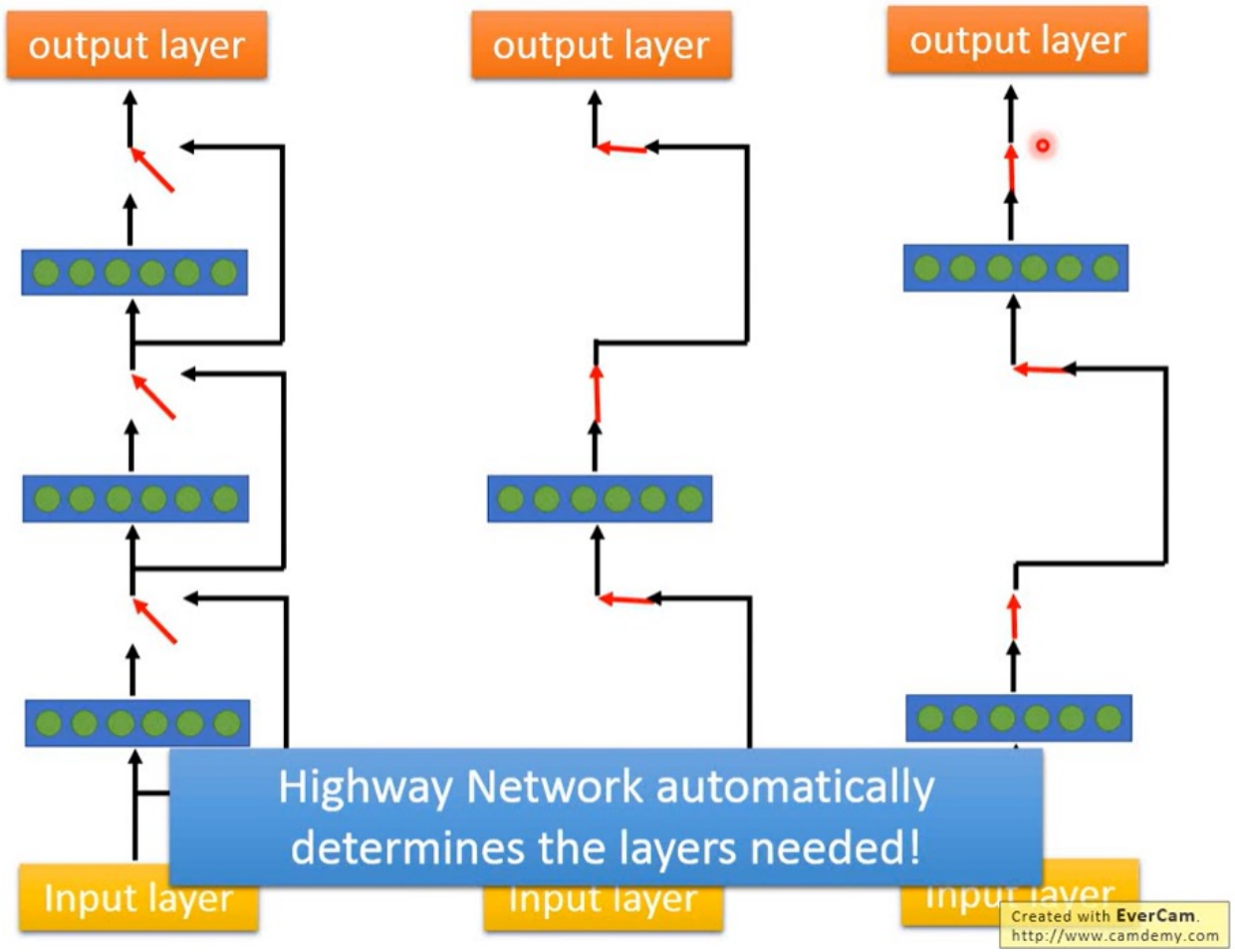


Highway network是前馈网络!

Highway Network

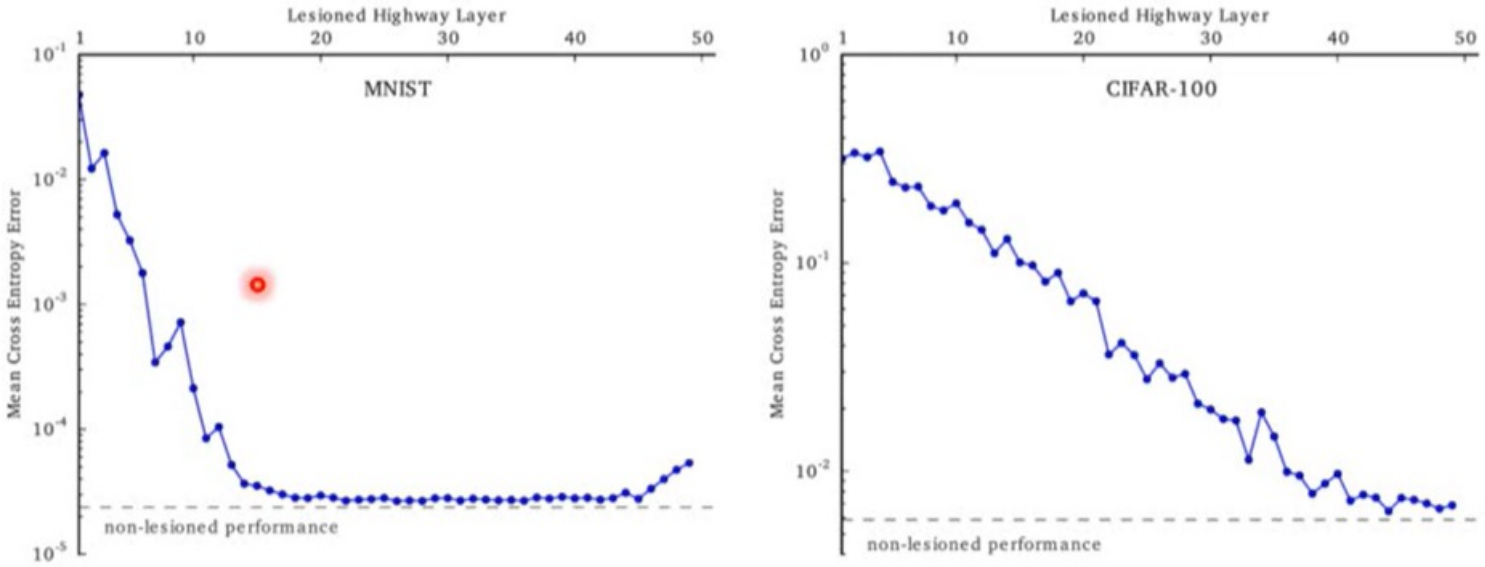


Rupesh Kumar Srivastava, Klaus Greff, Klaus Greff. Highway Networks. ICML. 2015



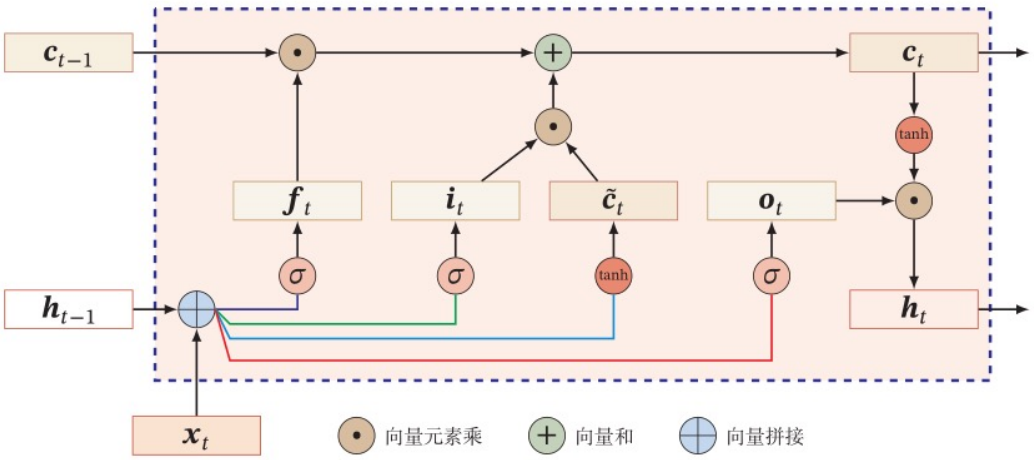
Rupesh Kumar Srivastava, Klaus Greff, Klaus Greff. Highway Networks. ICML. 2015

Highway Network



Rupesh Kumar Srivastava, Klaus Greff, Klaus Greff. Highway Networks. ICML. 2015

A LSTM Recap



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

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

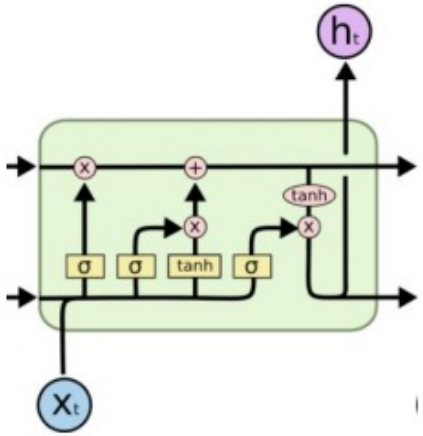
输出门 $\mathbf{o}_t = \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o),$

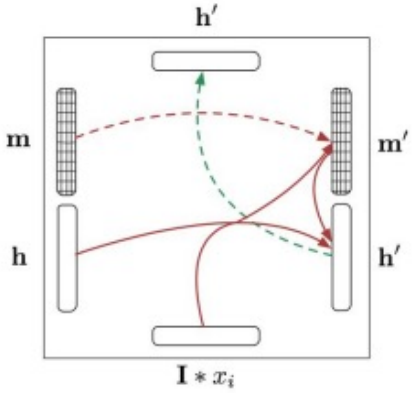
候选状态 $\tilde{\mathbf{c}}_t = \tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c)$

内部状态 $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,$

$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$

A LSTM Recap





Standard LSTM block

$$\mathbf{H} = \begin{bmatrix} Ix_i \\ \mathbf{h} \end{bmatrix}$$

$$\mathbf{g}^u = \sigma(\mathbf{W}^u\mathbf{H})$$

$$\mathbf{g}^f = \sigma(\mathbf{W}^f\mathbf{H})$$

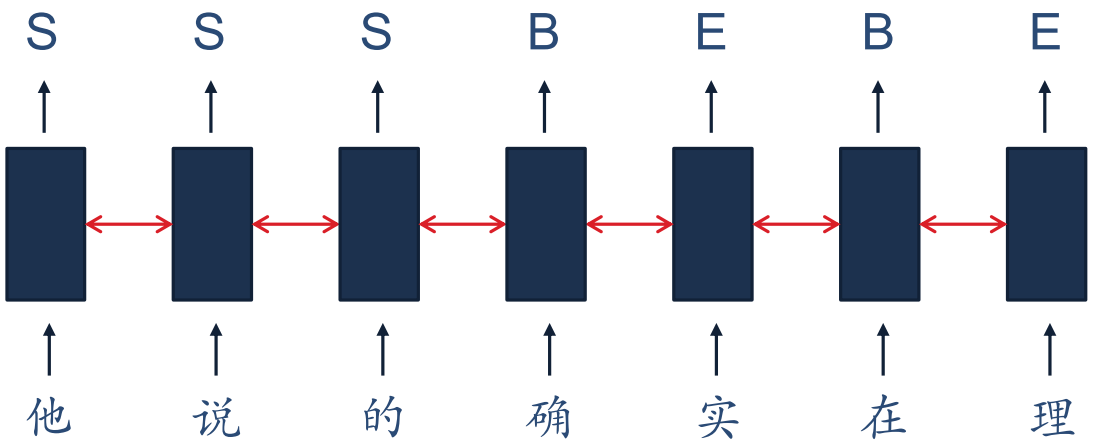
$$\mathbf{g}^o = \sigma(\mathbf{W}^o\mathbf{H})$$

$$\mathbf{g}^c = \tanh(\mathbf{W}^c\mathbf{H})$$

$$\mathbf{m}' = \mathbf{g}^f \odot \mathbf{m} + \mathbf{g}^u \odot \mathbf{g}^c$$

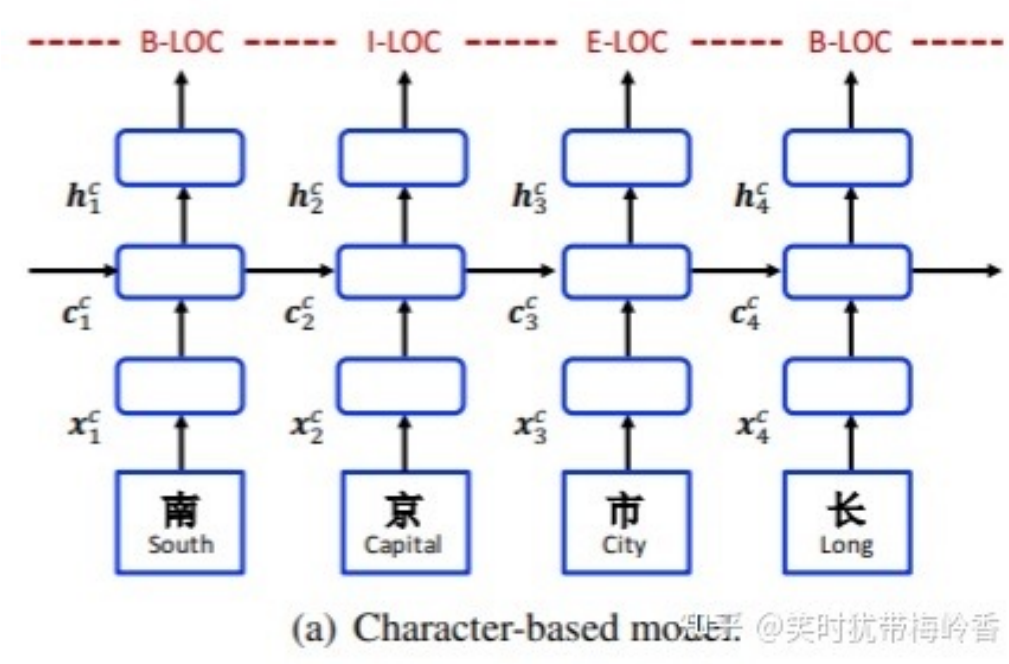
$$\mathbf{h}' = \tanh(\mathbf{g}^o \odot \mathbf{m}')$$

A Chinese Word Segmentation Recap



Lattice LSTM

- 中文命名实体识别(named entity recognition)-基于字符的模型



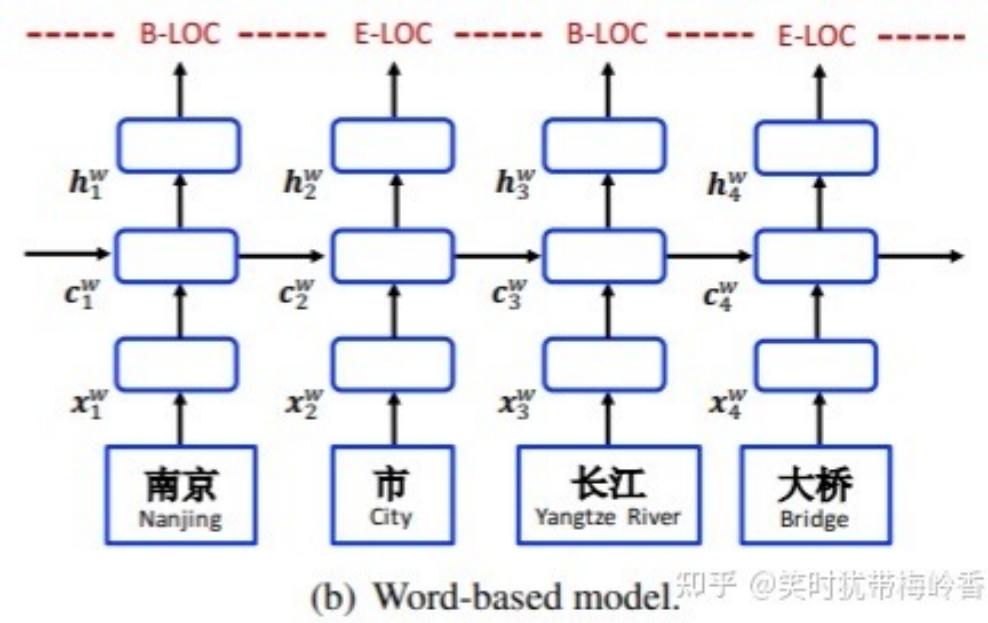
(a) Character-based model

Yue Zhang, Jie Yang. Chinese NER Using Lattice LSTM. ACL 2018

Lattice LSTM

https://arxiv.org/pdf/1805.02023.pdf

- 中文命名实体识别(named entity recognition)-基于词的模型



(b) Word-based model.

Lattice LSTM

https://arxiv.org/pdf/1805.02023.pdf

- 同时用到字符和词的信息

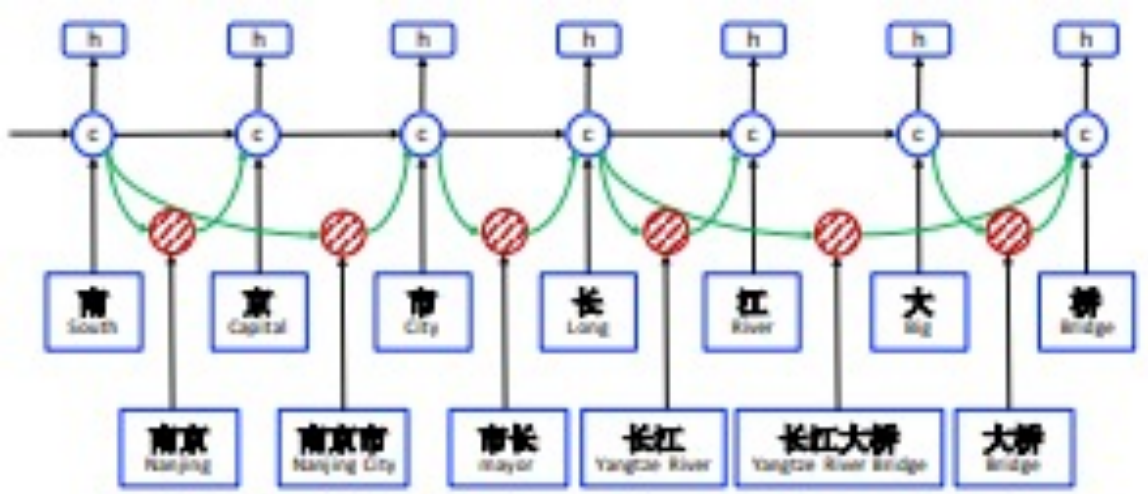
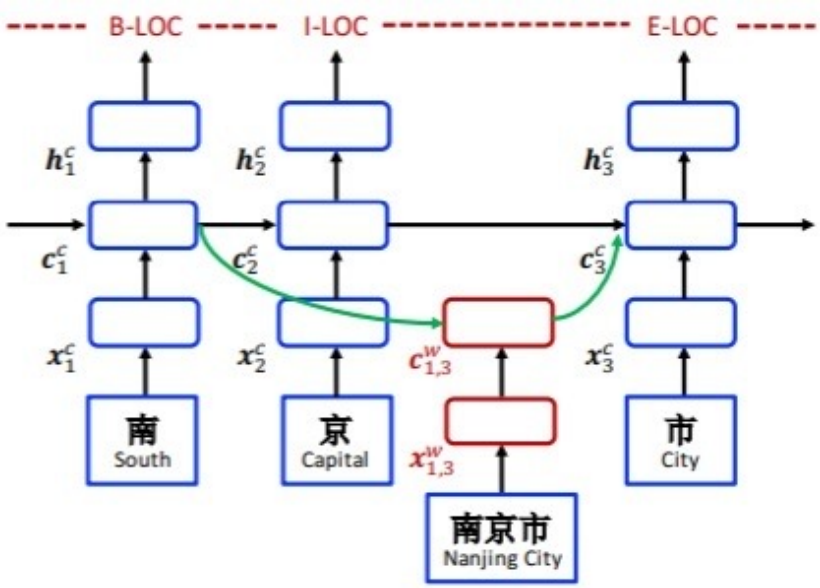


Figure 2: Lattice LSTM structure.

Lattice LSTM

https://arxiv.org/pdf/1805.02023.pdf

- 同时用到字符和词的信息



(c) Lattice model.

$$\begin{bmatrix} \mathbf{i}_j^c \\ \mathbf{o}_j^c \\ \mathbf{f}_j^c \\ \tilde{\boldsymbol{c}}_j^c \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}^{c\top} \begin{bmatrix} \mathbf{x}_j^c \\ \mathbf{h}_{j-1}^c \end{bmatrix} + \mathbf{b}^c \right)$$

$$\mathbf{c}_j^c = \mathbf{f}_j^c \odot \mathbf{c}_{j-1}^c + \mathbf{i}_j^c \odot \tilde{\boldsymbol{c}}_j^c$$

$$\mathbf{h}_j^c = \mathbf{o}_j^c \odot \tanh\left(\mathbf{c}_j^c\right)$$

$$\begin{bmatrix} \mathbf{i}_{b,e}^w \\ \mathbf{f}_{b,e}^w \\ \tilde{\boldsymbol{c}}_{b,e}^w \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}^{w\top} \begin{bmatrix} \mathbf{x}_{b,e}^w \\ \mathbf{h}_b^c \end{bmatrix} + \mathbf{b}^w \right)$$

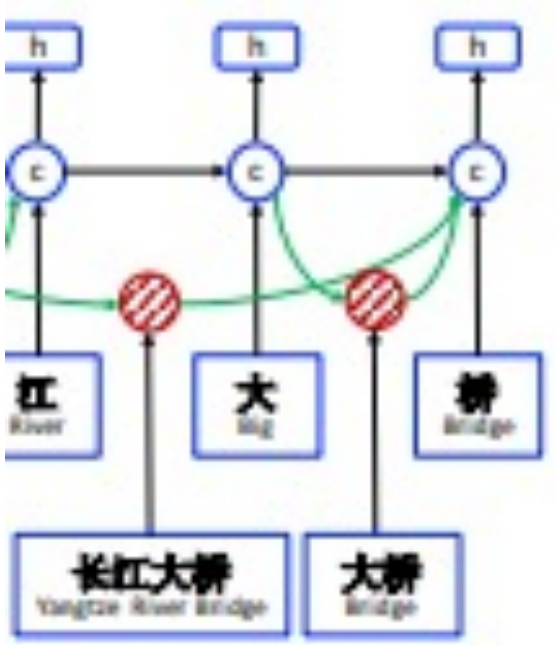
$$\mathbf{c}_{b,e}^w = \mathbf{f}_{b,e}^w \odot \mathbf{c}_b^c + \mathbf{i}_{b,e}^w \odot \tilde{\boldsymbol{c}}_{b,e}^w$$

$$\mathbf{c}_j^c = \sum_{b \in \left\{ b' | w_{b',j}^d \in \mathbb{D} \right\}} \boldsymbol{\alpha}_{b,j}^c \odot \mathbf{c}_{b,j}^w + \boldsymbol{\alpha}_j^c \odot \tilde{\boldsymbol{c}}_j^c$$

Lattice LSTM

https://arxiv.org/pdf/1805.02023.pdf

- 同时用到字符和词的信息



$$\mathbf{i}_{b,e}^{c} = \sigma\left(\mathbf{W}^{l\top}\begin{bmatrix}\mathbf{x}_{e}^{c}\\ \mathbf{c}_{b,e}^{w}\end{bmatrix} + \mathbf{b}^{l}\right)$$

$$\boldsymbol{\alpha}_{b,j}^{c} = \frac{\exp\left(\mathbf{i}_{b,j}^{c}\right)}{\exp\left(\mathbf{i}_{j}^{c}\right) + \sum_{b'\in\left\{b''|w_{b'',j}^{d}\in\mathbb{D}\right\}}\exp\left(\mathbf{i}_{b',j}^{c}\right)}$$

$$\boldsymbol{\alpha}_{j}^{c} = \frac{\exp\left(\mathbf{i}_{j}^{c}\right)}{\exp\left(\mathbf{i}_{j}^{c}\right) + \sum_{b'\in\left\{b''|w_{b'',j}^{d}\in\mathbb{D}\right\}}\exp\left(\mathbf{i}_{b',j}^{c}\right)}$$

Lattice LSTM

https://arxiv.org/pdf/1805.02023.pdf

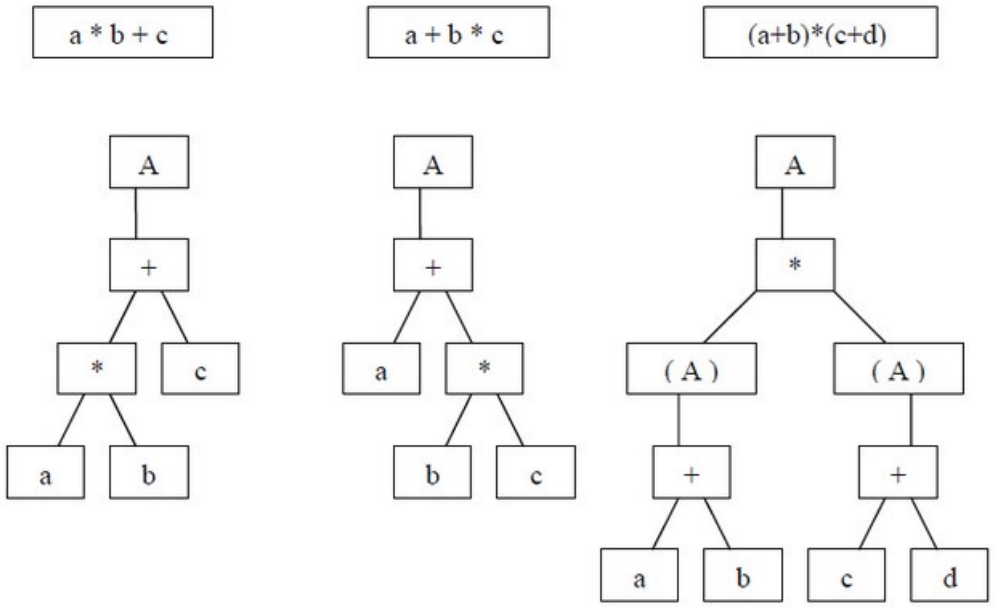
- 优点
  - 同时利用了字符和词维度的信息并且以合理的方式融入了循环神经网络
- 缺点
  - 计算性能低下，不能batch并行化
  - 可迁移性差

扩展到图结构

扩展到图结构

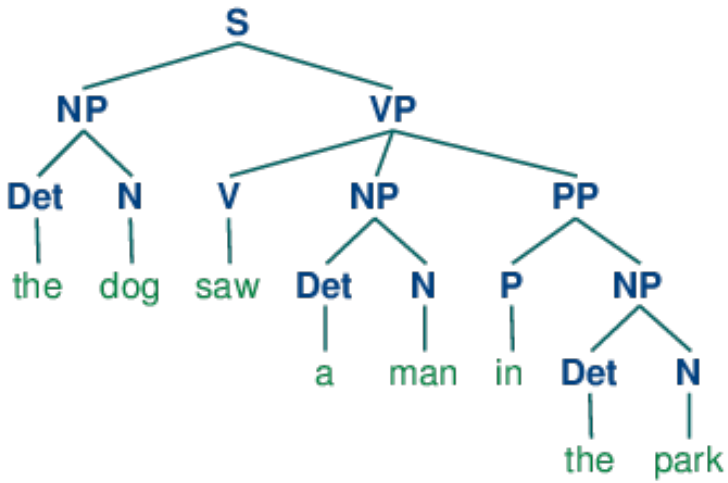
树结构

- 程序语言的句法结构



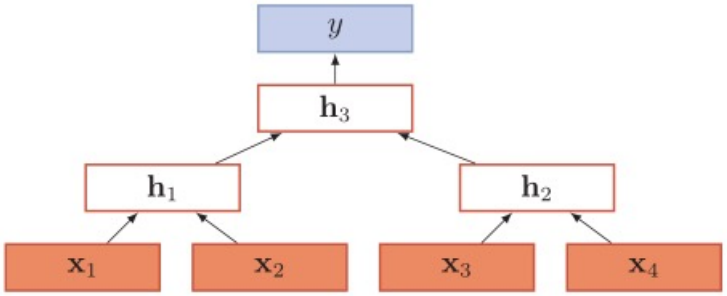
树结构

- 自然语言的句法结构



递归神经网络Recursive Neural Network

- 递归神经网络在一个有向图无循环图上共享一个组合函数



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

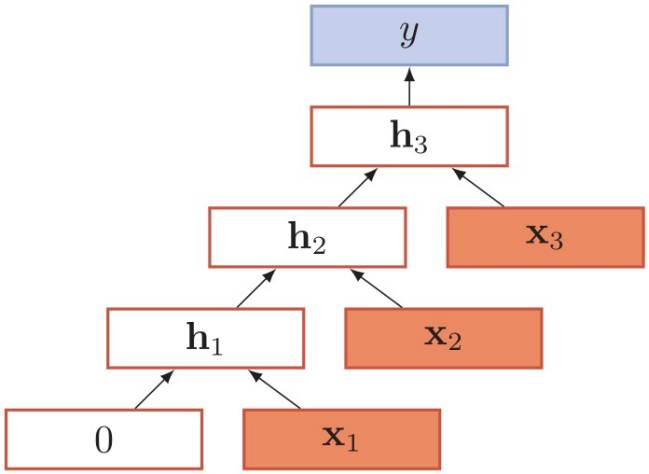
$$\mathbf{h}_1 = f(W \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \mathbf{b}),$$

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

$$\mathbf{h}_3 = f(W \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} + \mathbf{b}),$$

递归神经网络

- 退化为循环神经网络



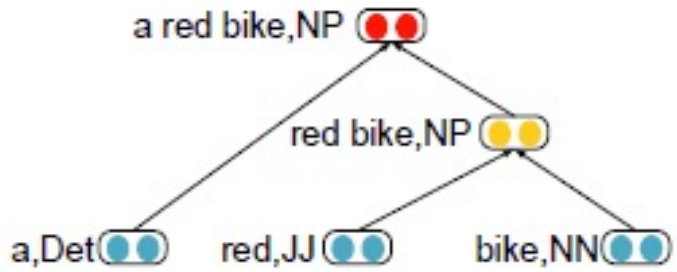
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t),$$

递归神经网络

给定一个语法树(Constituency Path)，

$p_2 \rightarrow ap_1$,

$p_1 \rightarrow bc$.



$$\mathbf{p}_1 = f(\mathbf{W}\begin{bmatrix}\mathbf{b}\\ \mathbf{c}\end{bmatrix}),$$

$$\mathbf{p}_2 = f(\mathbf{W}\begin{bmatrix}\mathbf{a}\\ \mathbf{p}_1\end{bmatrix}).$$

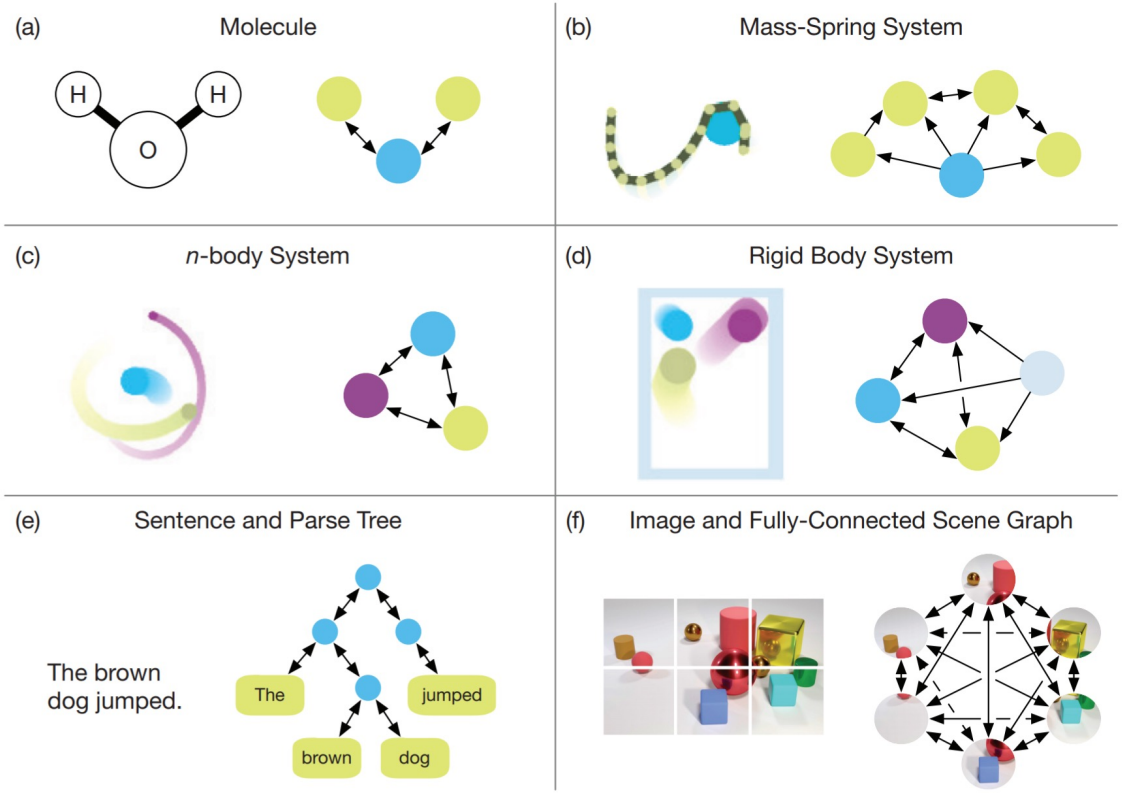
Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. EMNLP 2013

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。

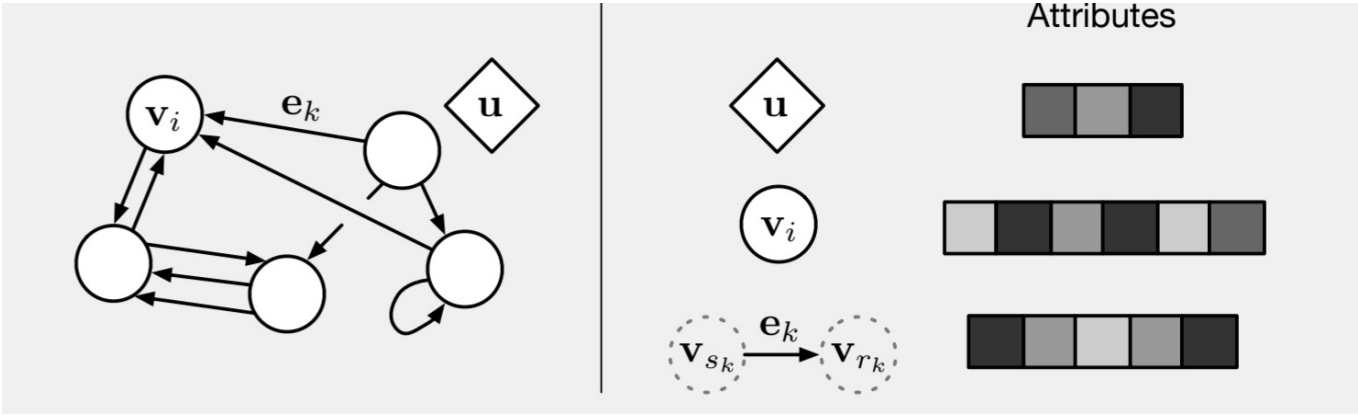
图数据

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



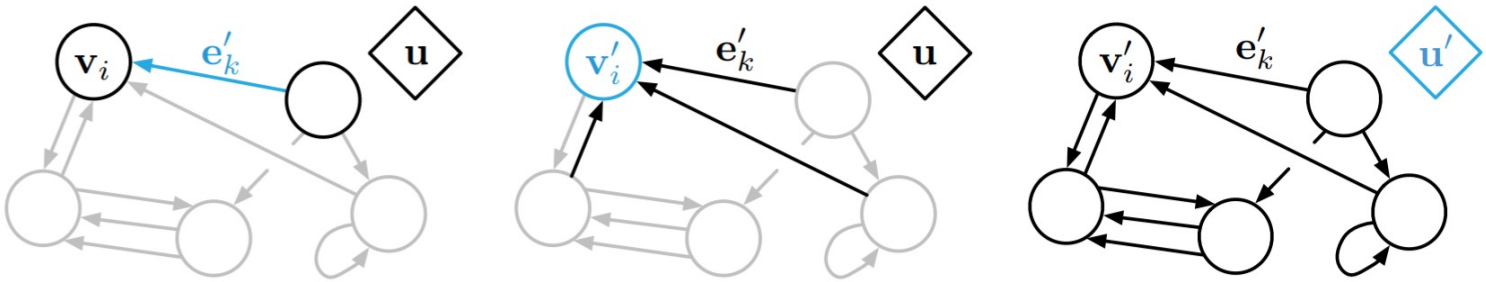
图网络

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

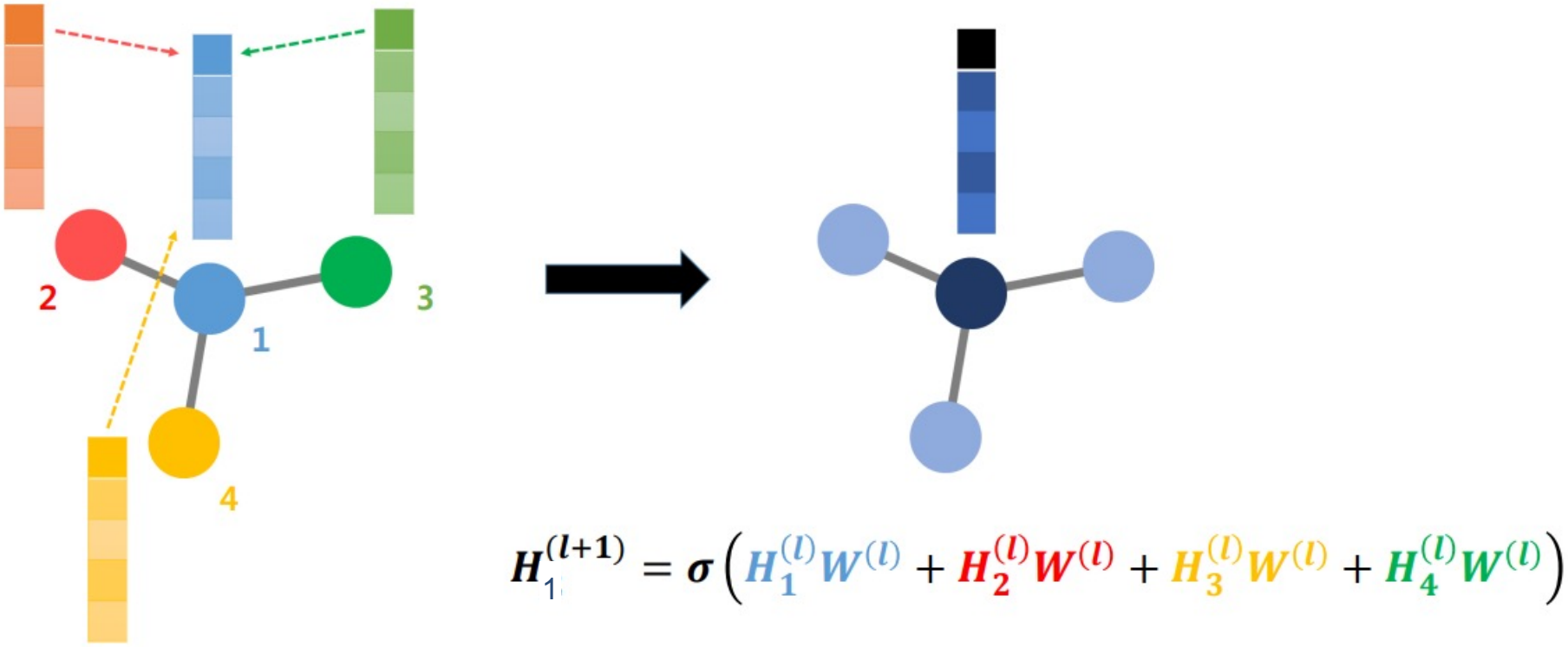
$$\mathbf{m}_t^{(v)} = \sum_{u \in N(v)} f\left(\mathbf{h}_{t-1}^{(v)}, \mathbf{h}_{t-1}^{(u)}, \mathbf{e}^{(u,v)}\right)$$

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

  - 读出函数

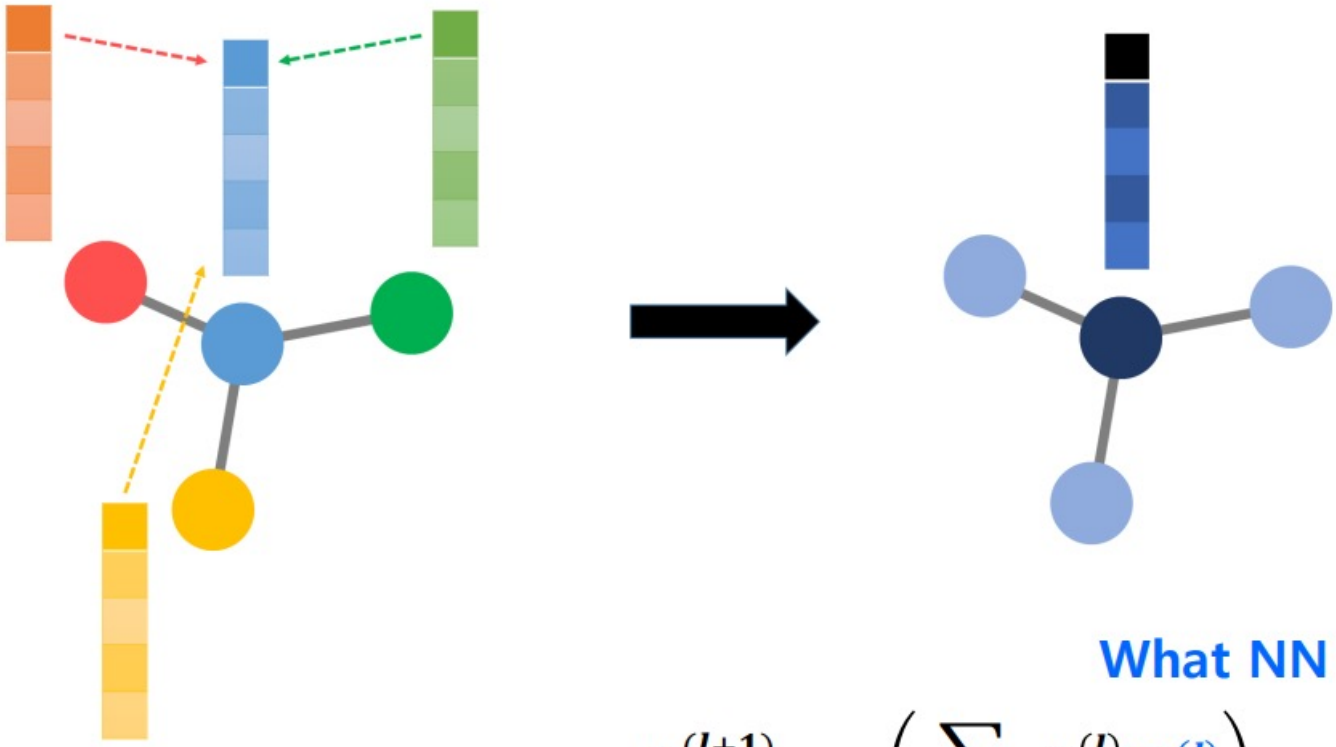
$$\mathbf{y}_t = g\left(\{\mathbf{h}_T^{(v)} | v \in \mathcal{V}\}\right)$$

Graph Convolutional Network(GCN)



$$H_1^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + H_4^{(l)}W^{(l)}\right)$$

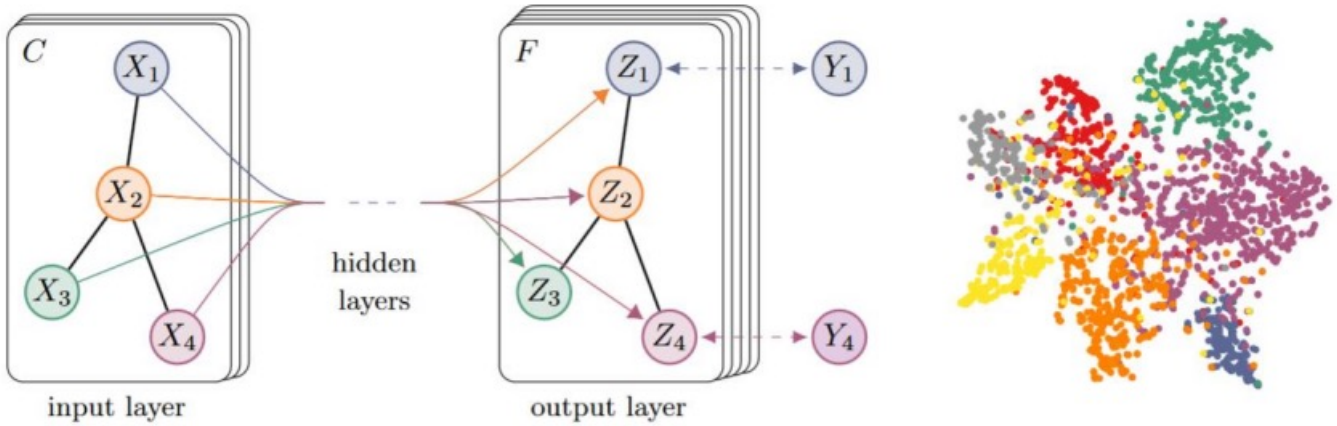
Graph Convolutional Network



What NN learns

$$H_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} H_j^{(l)} W^{(l)}\right) = \sigma\left(A H_j^{(l)} W^{(l)}\right)$$

Graph Convolutional Network



(a) Graph Convolutional Network

(b) Hidden layer activations

Figure 1: *Left*: Schematic depiction of multi-layer Graph Convolutional Network (GCN) for semi-supervised learning with C input channels and F feature maps in the output layer. The graph structure (edges shown as black lines) is shared over layers, labels are denoted by Y_i. *Right*: t-SNE (Maaten & Hinton, 2008) visualization of hidden layer activations of a two-layer GCN trained on the Cora dataset (Sen et al., 2008) using 5% of labels. Colors denote document class.

- Classification of nodes of citation networks and a knowledge graph
- $L = \sum_{l \in y_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$

Kipf, Thomas N., and Max Welling.
"Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016)

循环网络应用

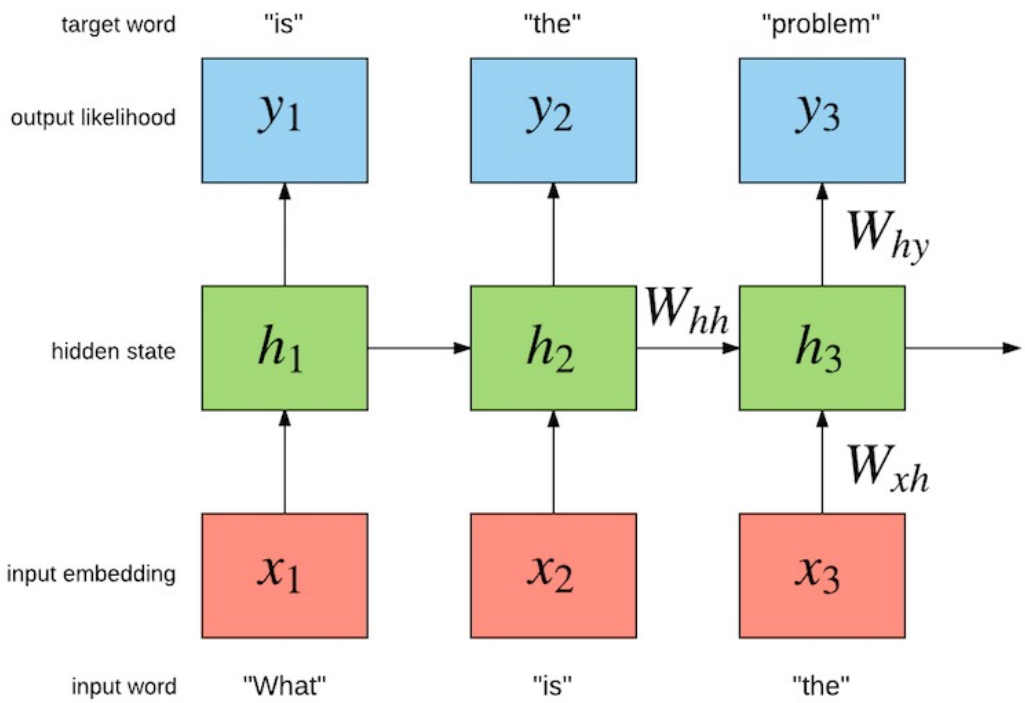
语言模型(Language Modeling)

- 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
h_3
W_{hh}
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旪
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

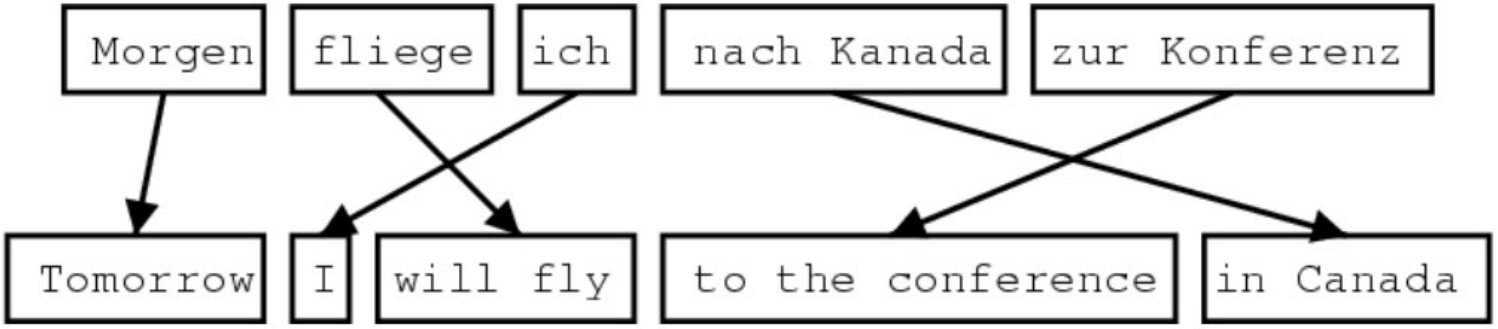
作诗

http://jiuge.thunlp.org/

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?

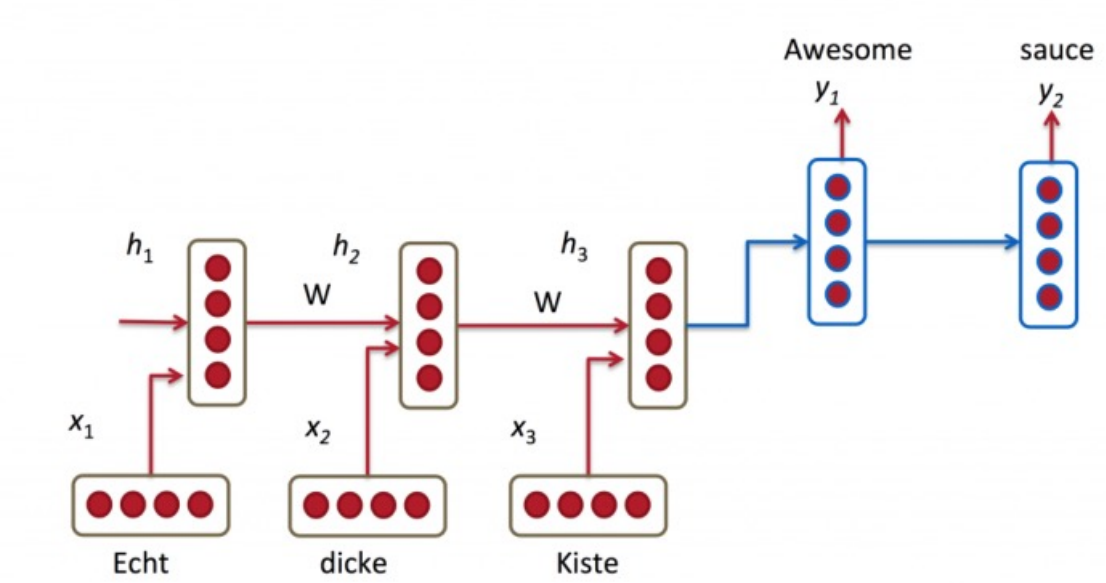
传统统计机器翻译

- 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 $p(f|e) = \prod_j p(f_j|e(A_j))$
  - p(e)：语言模型



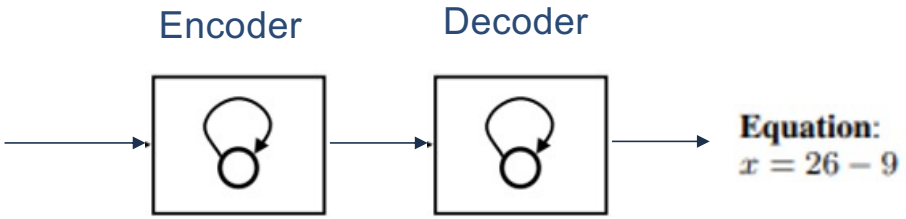
基于循环神经网络的机器翻译(Machine Translation)

- 一个RNN用来编码, Encoder
- 另一个RNN用来解码, Decoder

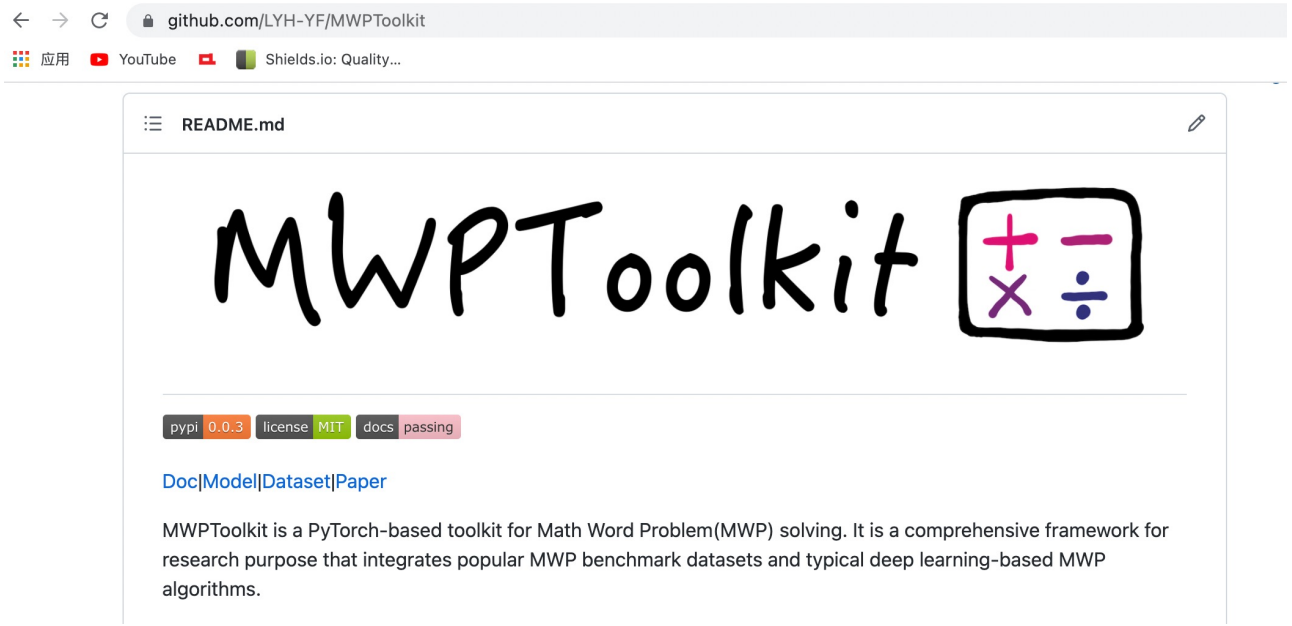


自动求解数学应用题(Math Word Problem Solving)

Single Equation Generation:
Paco had 26 salty cookies and 17 sweet cookies. He ate 14 sweet cookies and 9 salty cookies. How many salty cookies did Paco have left?



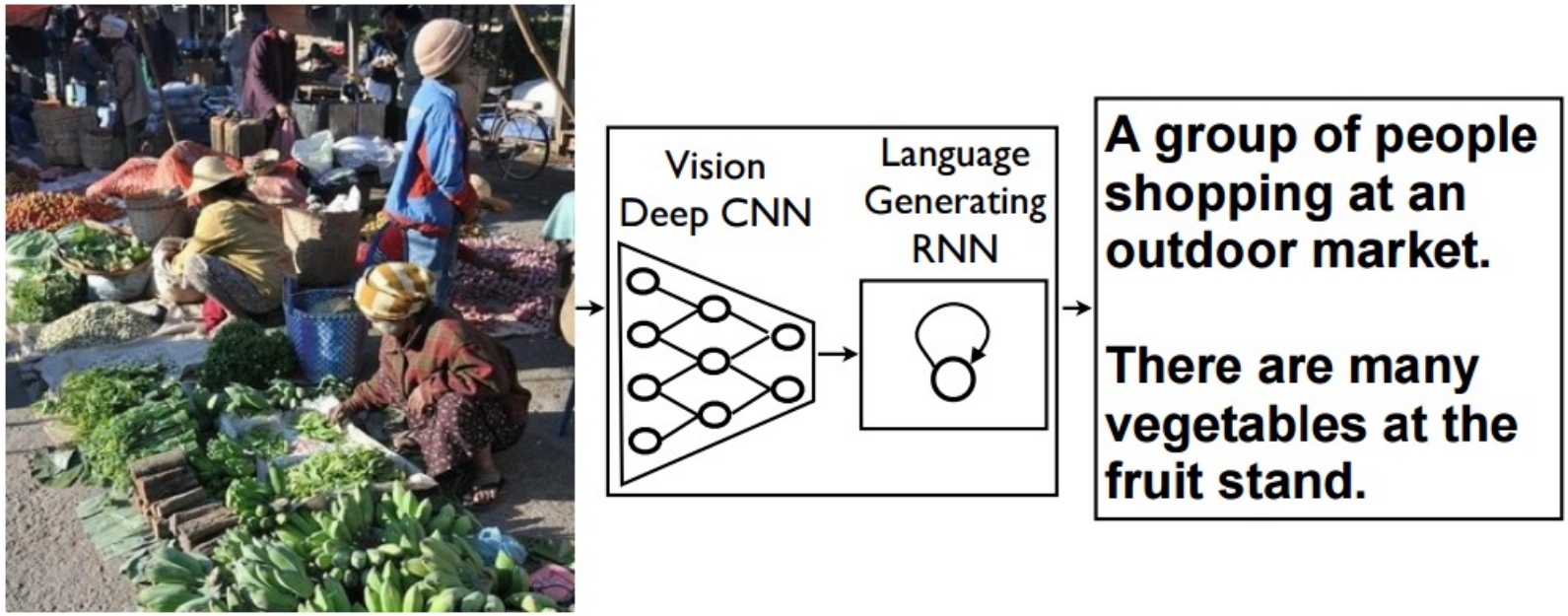
自动求解数学应用题(Math Word Problem Solving)



看图说话(Image Caption)

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

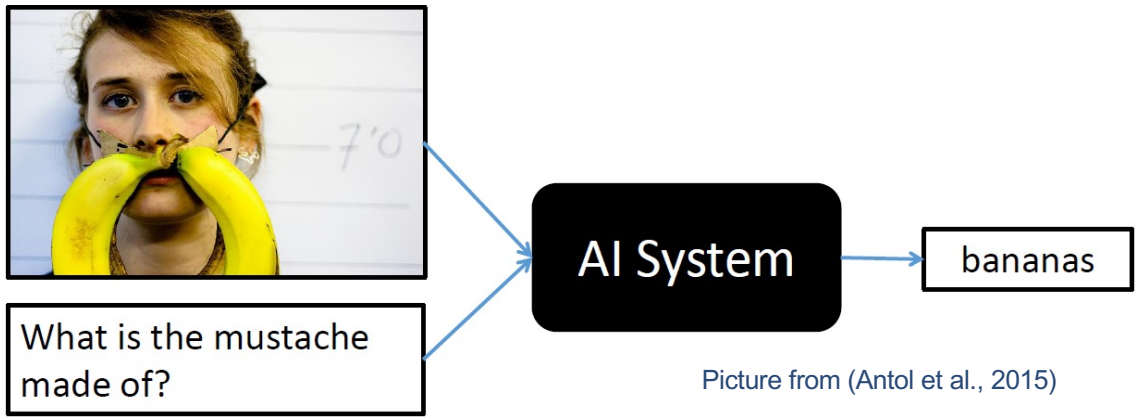
看图说话(Image Caption)



Visual Question Answering (VQA)

Demo Website

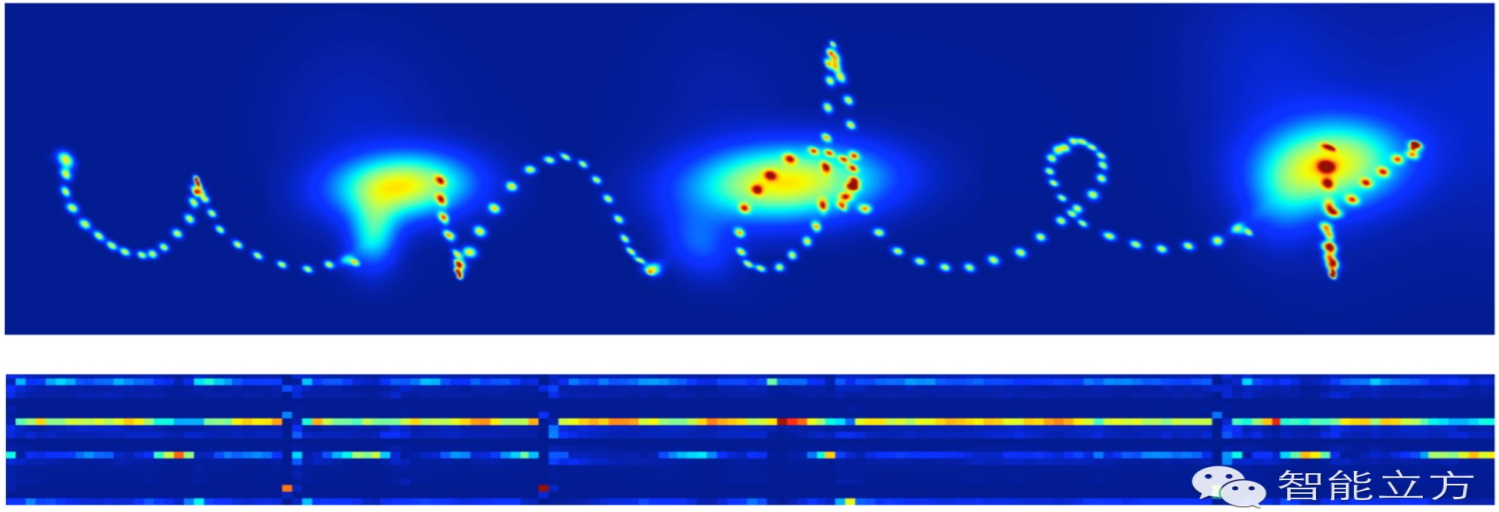
VQA: Given an image and a natural language question about the image, the task is to provide an accurate natural language answer



Picture from (Antol et al., 2015)

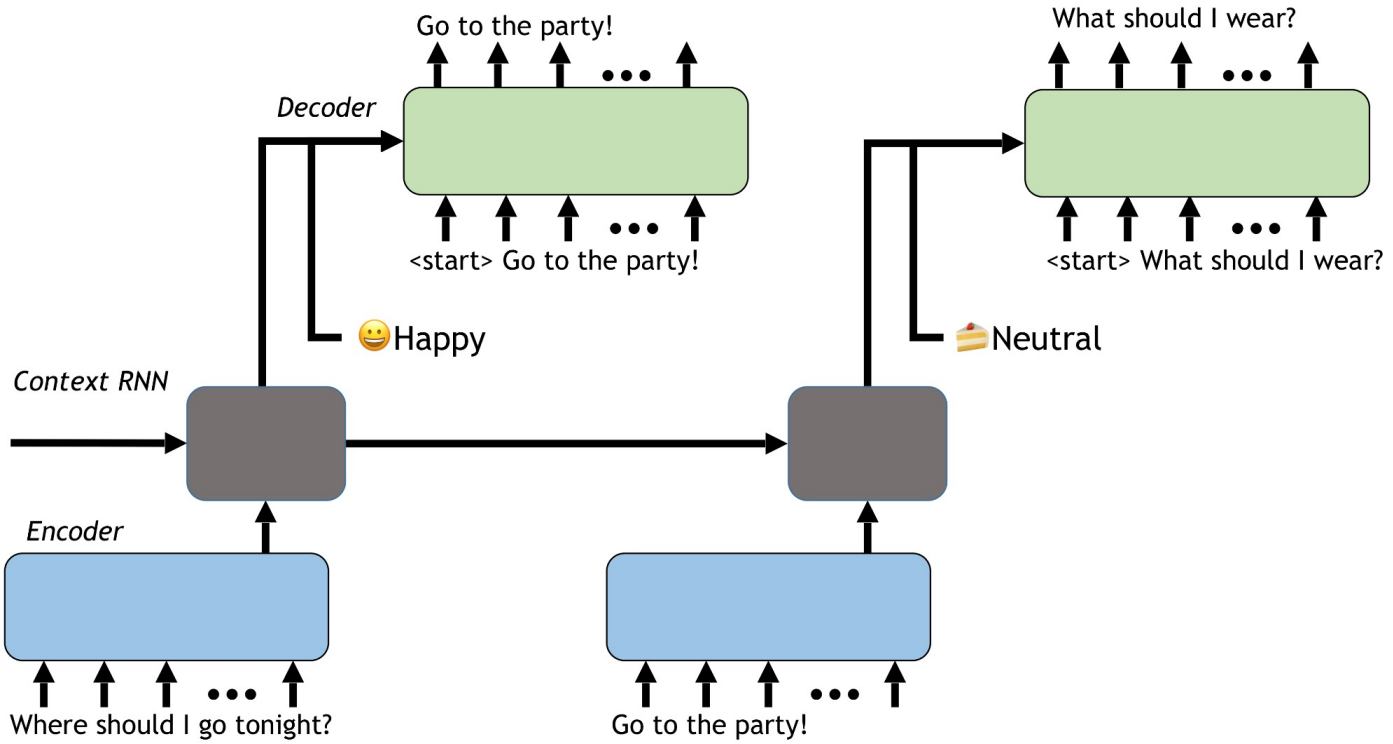
写字

- 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

教学内容

- 典型的循环神经网络
  - Stacked LSTM,
  - Bi-LSTM
  - Highway Network (前馈网络！)
  - Grid LSTM
  - Lattice-LSTM
- 扩展到图结构
  - RecNN
  - GCN
- 循环神经网络的应用
  - Machine Translation, Language Modeling, Image Caption, …