《物理与人工智能》

12. 词表示与递归神经网络

授课教师： 马滟青

2025/10/20（第六周）

鸣谢：基于计算机学院《人工智能引论》课程组幻灯片

北京大學

词表示（Word Representation）

PEKING UNIVERSITY

- **分布式表示** (distributed representation)
  - ✓ **理论基础**：词的语义由其上下文决定! 上下文相似的词，其语义也相似。

"He wrote a book."

he	[-0.34, -0.08, 0.02, -0.18, 0.22, ...]
wrote	[-0.27, 0.40, 0.00, -0.65, -0.15, ...]
a	[-0.12, -0.25, 0.29, -0.09, 0.40, ...]
book	[-0.23, -0.16, -0.05, -0.57, ...]

词表示（Word Representation）

- **分布式表示** (distributed representation)
  - ✓ 基于矩阵的分布表示
  - ✓ 基于聚类的分布表示
  - ✓ 基于神经网络的分布表示
  - **核心思想：**
    - ➢ 选择一种方式描述上下文
    - ➢ 选择一种模型刻画目标词与其上下文之间的关系

word2vec

- **CBOW模型**

① 对每个词随机初始化为一个$1 \times V$向量

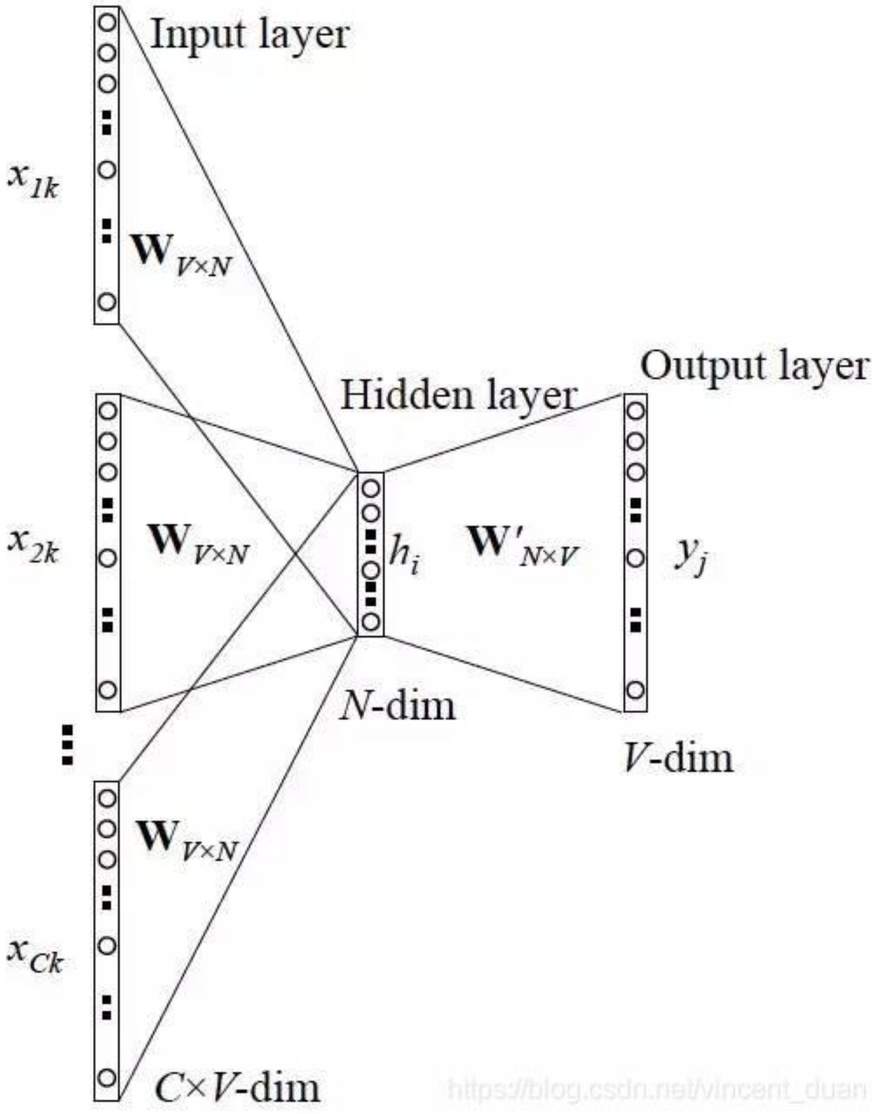
② 把当前词的上下文词语的向量各乘同一个矩阵$\boldsymbol{W}$（周围词向量矩阵）得到各自的$1 \times N$向量

③ 将这些$1 \times N$向量取平均为一个$1 \times N$向量

④ 将这个$1 \times N$向量乘矩阵$\boldsymbol{W}'$ (中心词向量矩阵), 变成一个$1 \times V$向量

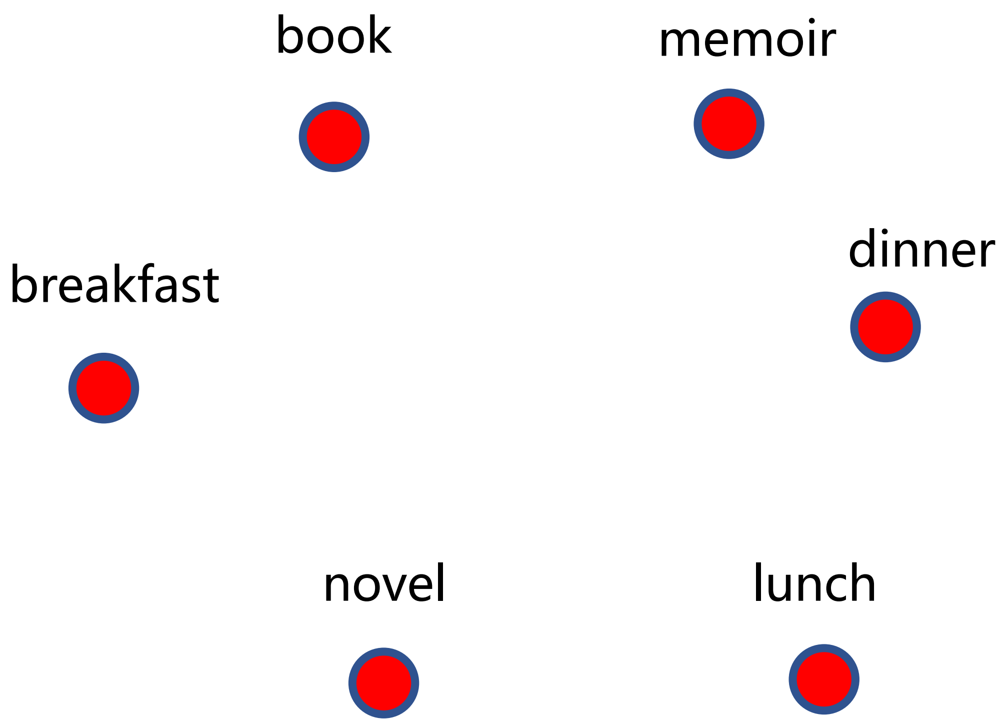
⑤ 做Softmax分类，与真实标签$1 \times V$向量计算交叉熵损失

⑥ 每次前向传播之后反向传播误差，调整矩阵$\boldsymbol{W}$和$\boldsymbol{W}'$的值



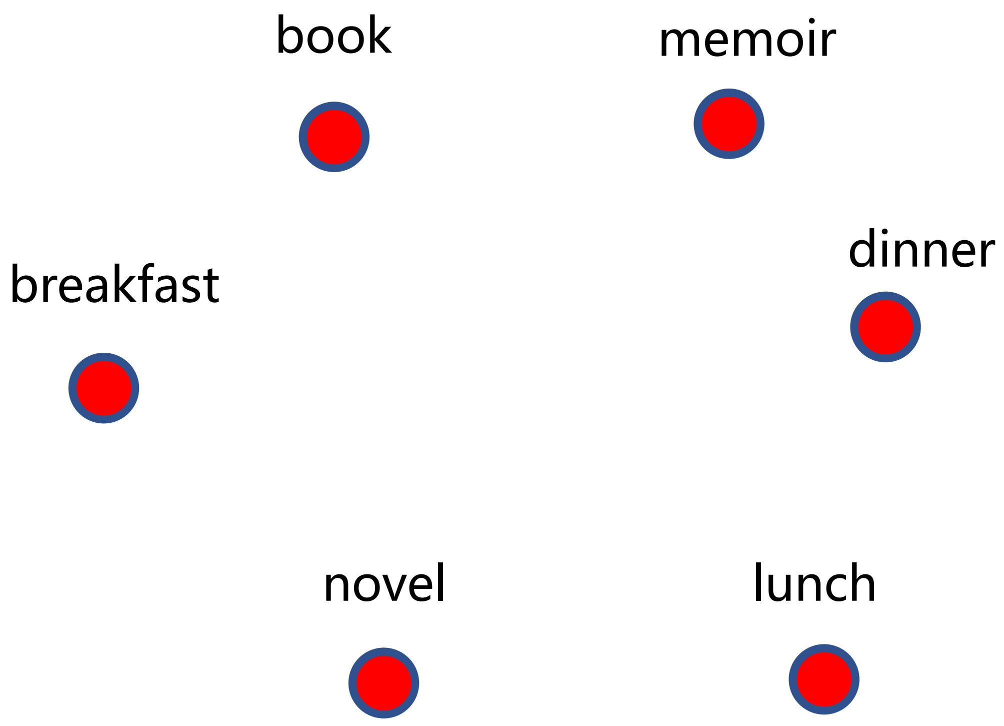
Efficient Estimation of Word Representations in Vector Space. ICLR 2013

word2vec



独特表示的词特征空间示例

word2vec



自监督学习后的word2vec词特征空间示例

word2vec

自监督学习后的word2vec词特征空间示例

word2vec——词向量示例

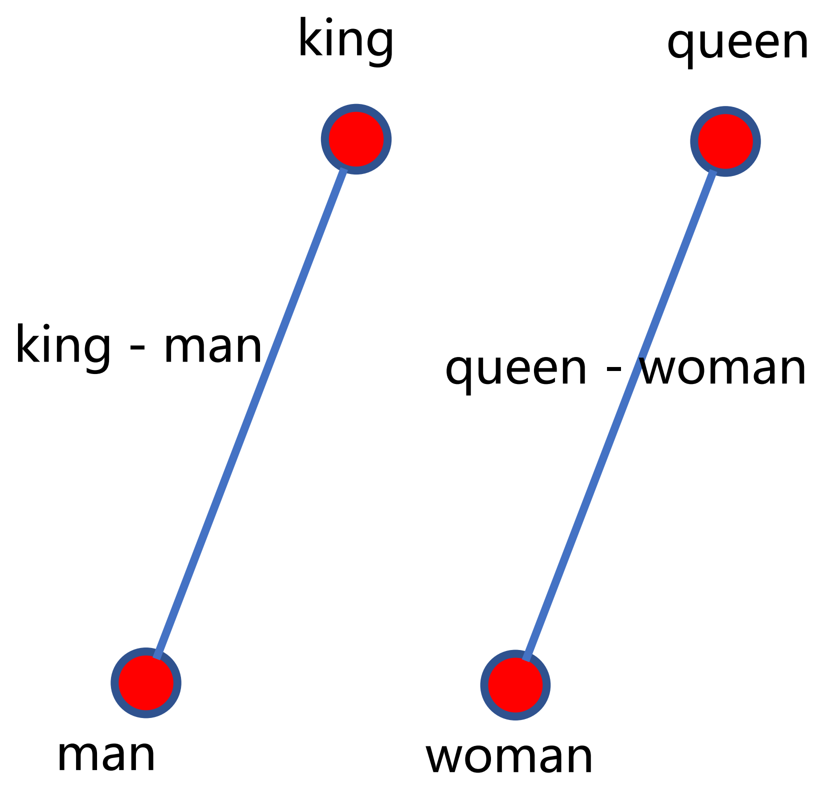
```
>>> words["city"]
array([ 0.231087, -0.238098,  0.584713, -0.524351,  0.40278 ,  0.148448,
        0.386096, -0.493994, -0.198922, -0.411161,  0.556962,  0.220978,
       -0.304637, -0.499713, -0.092555,  0.262613,  0.752704,  0.463667,
        0.054477,  0.155809, -0.195134, -0.009269,  0.378139, -0.651306,
       -0.029372, -0.563472,  0.024709,  0.366842, -0.476904, -0.42565 ,
       -0.094642, -0.052822,  0.124612,  0.296046, -0.244881,  0.195957,
        0.223666,  0.064116,  0.577874,  0.083096, -0.378262,  0.196044,
       -0.220993, -0.630213, -0.311214,  0.435611,  0.351486,  0.342794,
       -0.229961, -0.157521,  0.204315,  0.253944, -0.562277,  0.534482,
       -0.4158  ,  0.120161,  0.649395, -0.227012, -0.130488, -0.332326,
        0.691952, -0.400436,  0.410125,  0.026237, -0.408483,  0.188236,
        0.130957, -0.320686,  0.225932, -0.171665, -0.335107, -0.009982,
        0.680831, -0.023788, -0.165798,  0.345986, -0.232295,  0.021137,
        0.08515 , -0.24387 , -0.142469, -0.058325,  0.086046, -0.173068,
        0.198108,  0.009103,  0.381725,  0.095911,  0.317972, -0.10012 ,
        0.143178,  0.106724, -0.419844, -0.175785, -0.251805,  0.211927,
        0.411175,  0.317378,  0.450316, -0.252661])
```

```
>>> distance(words["book"],words["book"])
0
>>> distance(words["book"],words["breakfast"])
0.6351827719357863
>>> distance(words["book"],words["novel"])
0.3436623421719047
>>> distance(words["lunch"],words["breakfast"])
0.2006302059301045
```

```
>>> closest_words(words["book"])[:6]
['book', 'books', 'essay', 'memoir', 'essays', 'novella']
>>> closest_words(words["lunch"])[:6]
['lunch', 'dinner', 'lunches', 'snack', 'meal', 'brunch']
```

```
>>> closest_word(words["woman"] + words["king"] - words["man"])
'queen'
>>> closest_word(words["england"] + words["paris"] - words["france"])
'london'
```

```
>>> closest_word(words["japan"] + words["beijing"] - words["china"])
'tokyo'
>>> closest_word(words["hospital"] + words["teacher"] - words["school"])
'nurse'
>>> closest_word(words["hospital"] + words["student"] - words["school"])
'hospital'
```



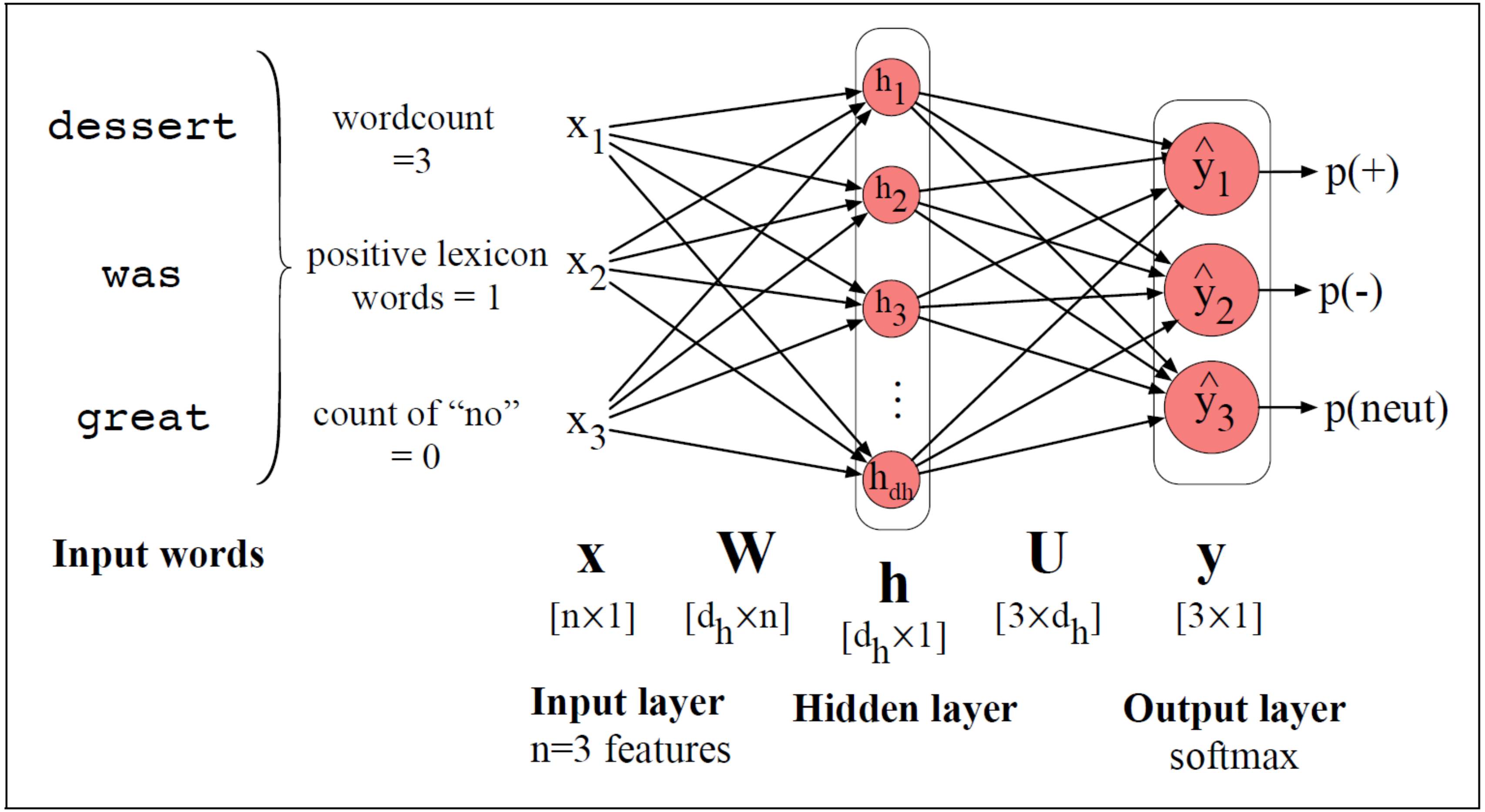

```
>>> closest_words(words["hospital"] + words["student"] - words["school"])
['hospital', 'patient', 'icu', 'nurse', 'nurses', 'transplant', 'gyn', 'neurosurgery', 'inpatient', 'medical']
```

基于神经网络的自然语言处理方法

文本情感分类应用：基于人工设计文本特征

- $\mathbf{x}_i$是人工设计特征，可以是标量

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, ... \mathbf{x}_N]$$
$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$\mathbf{z} = \mathbf{U}\mathbf{h}$$
$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$$



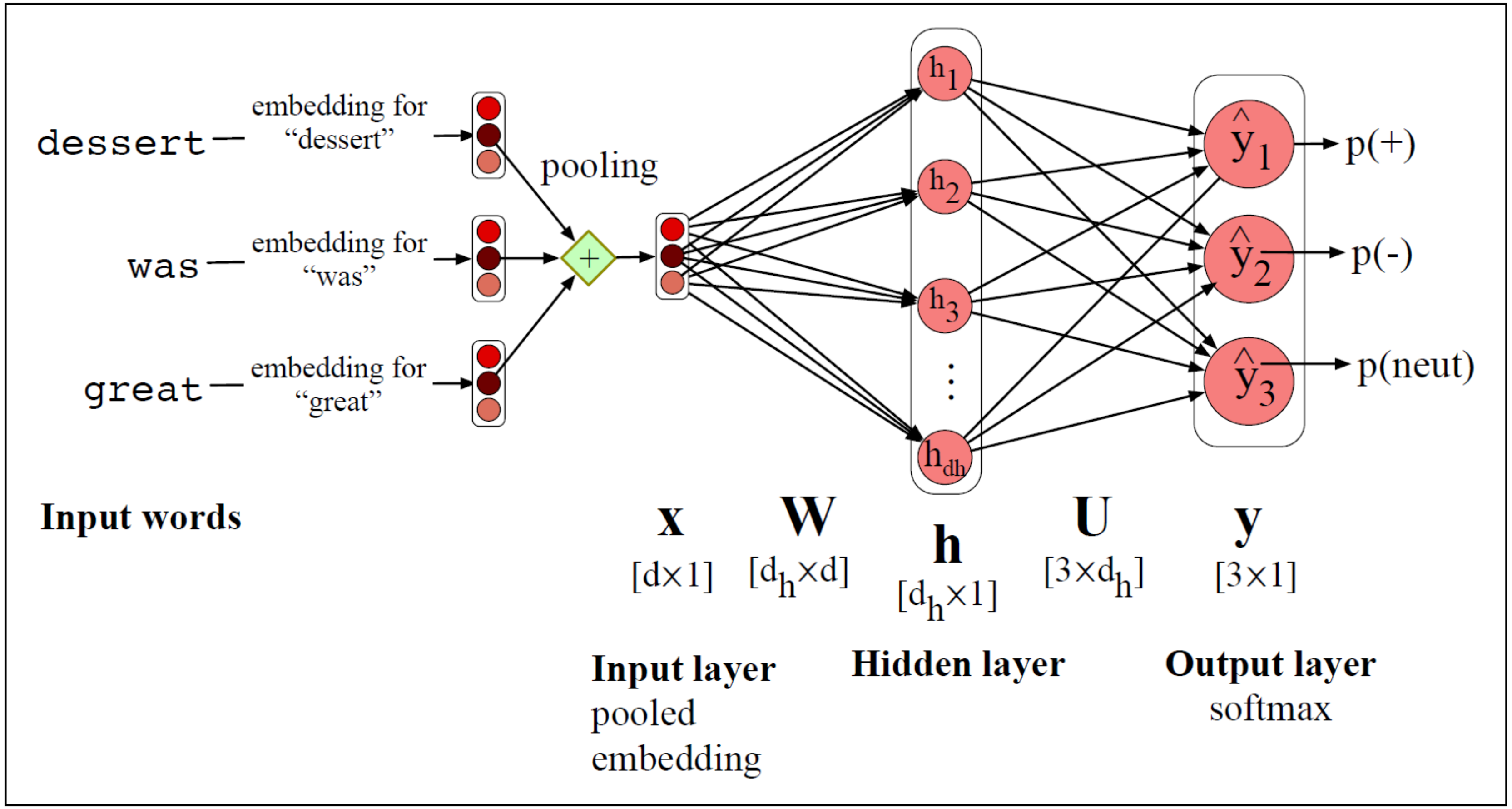
基于神经网络的自然语言处理方法

文本情感分类应用：基于word2vec/GloVe的文本特征

- 文本包含n个词w_i
- $\mathbf{e}(w_i)$: word2vec词嵌入

$$
\begin{aligned}
\mathbf{x} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{e}(w_i) \\
\mathbf{h} &= \sigma(\mathbf{W}\mathbf{x}+\mathbf{b}) \\
\mathbf{z} &= \mathbf{U}\mathbf{h} \\
\hat{\mathbf{y}} &= \mathrm{softmax}(\mathbf{z})
\end{aligned}
$$

sum, avg, concat等均可

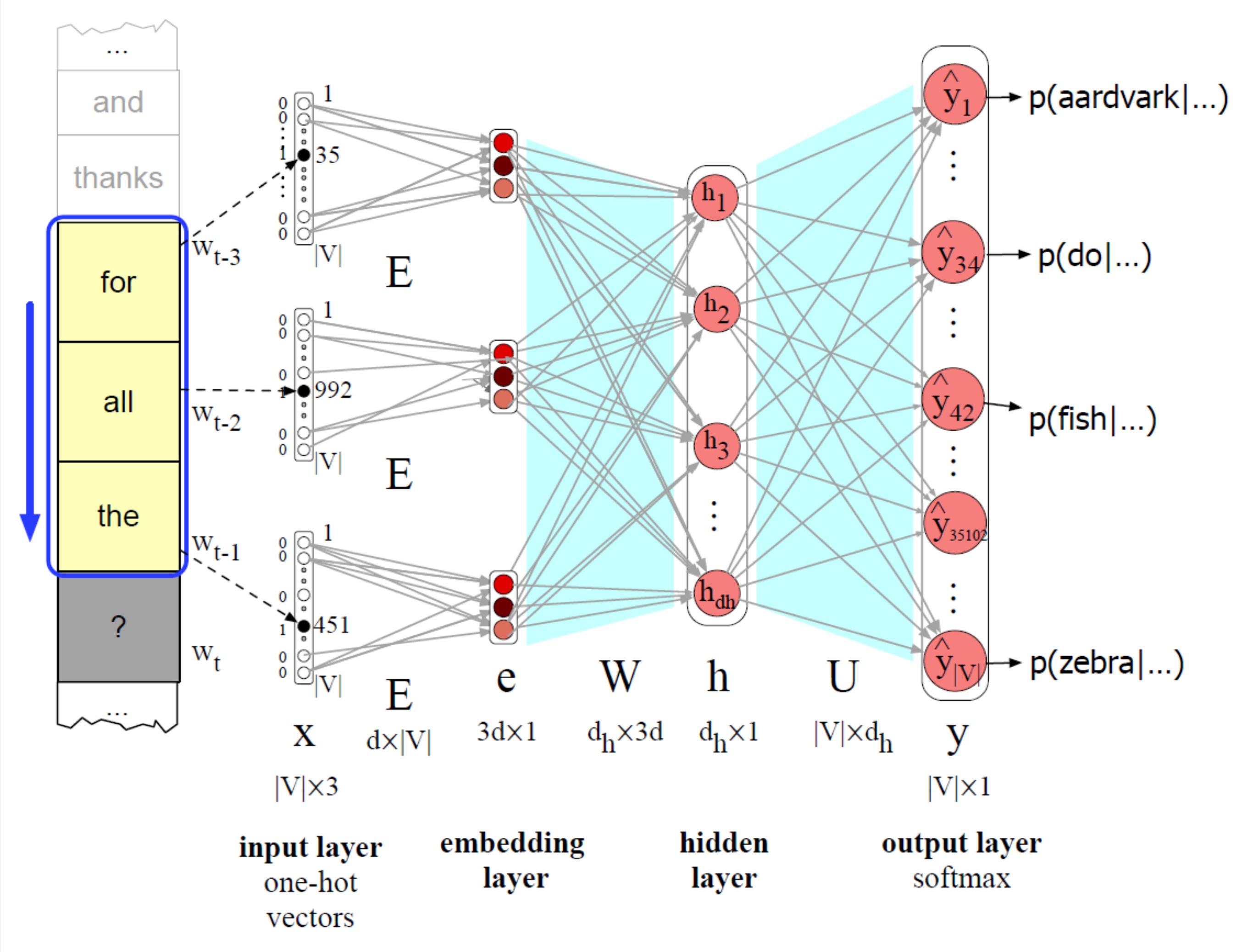


基于神经网络的自然语言处理方法

前馈神经语言模型：基于前词上下文预测接下来的词

- 输入独热向量/或嵌入向量
- 给定窗口大小(如3)

$$
\begin{aligned}
\mathbf{e} &= [\mathbf{E}\mathbf{x}_{t-3}; \mathbf{E}\mathbf{x}_{t-2}; \mathbf{E}\mathbf{x}_{t-1}] \\
\mathbf{h} &= \sigma(\mathbf{W}\mathbf{e} + \mathbf{b}) \\
\mathbf{z} &= \mathbf{U}\mathbf{h} \\
\hat{\mathbf{y}} &= \mathrm{softmax}(\mathbf{z})
\end{aligned}
$$

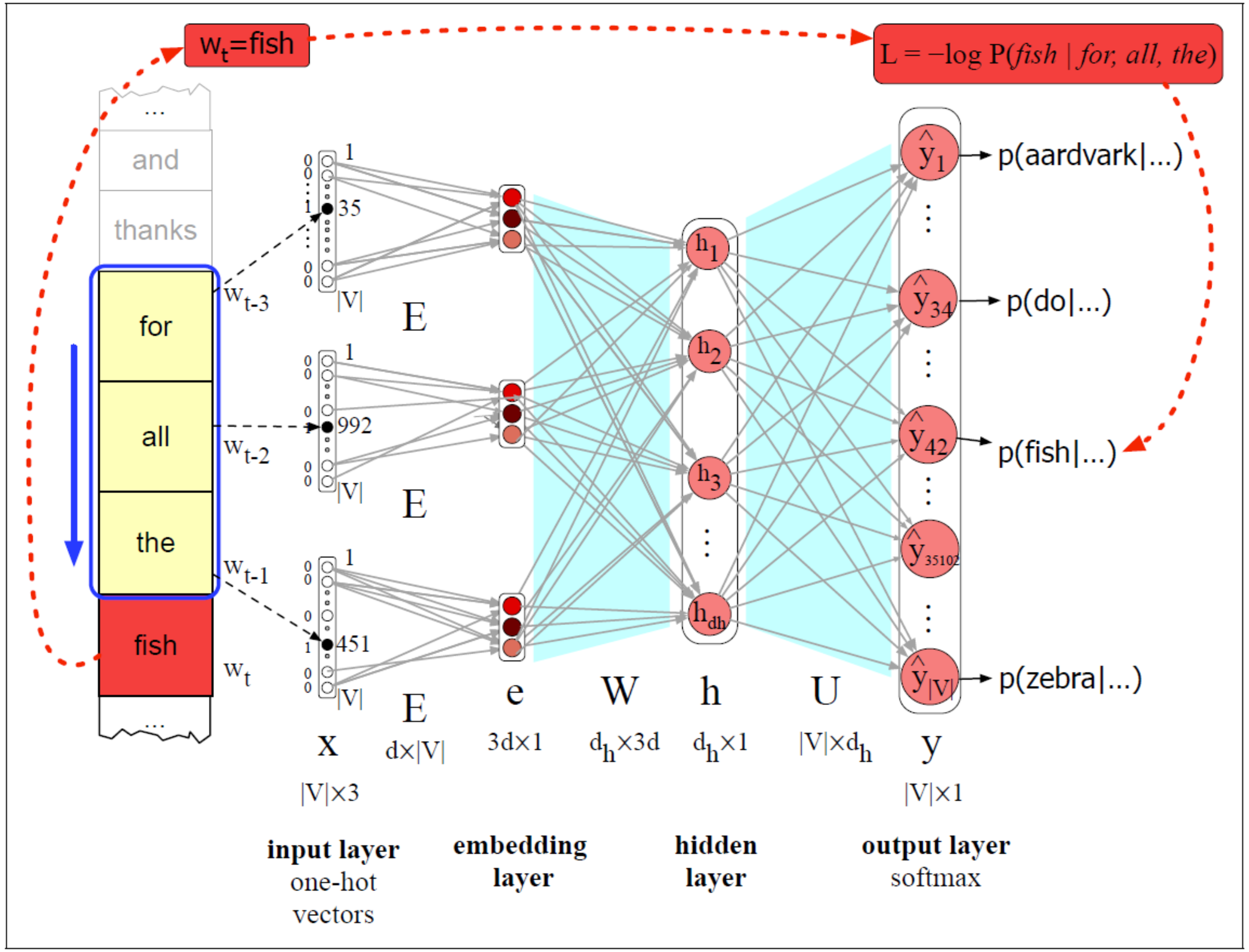


基于神经网络的自然语言处理方法

前馈神经语言模型：基于前词上下文预测接下来的词

- 输入独热向量/或嵌入向量
- 给定窗口大小(如3)

$$
\begin{aligned}
\mathbf{e} &= [\mathbf{E}\mathbf{x}_{t-3}; \mathbf{E}\mathbf{x}_{t-2}; \mathbf{E}\mathbf{x}_{t-1}] \\
\mathbf{h} &= \sigma(\mathbf{W}\mathbf{e} + \mathbf{b}) \\
\mathbf{z} &= \mathbf{U}\mathbf{h} \\
\hat{\mathbf{y}} &= \mathrm{softmax}(\mathbf{z})
\end{aligned}
$$

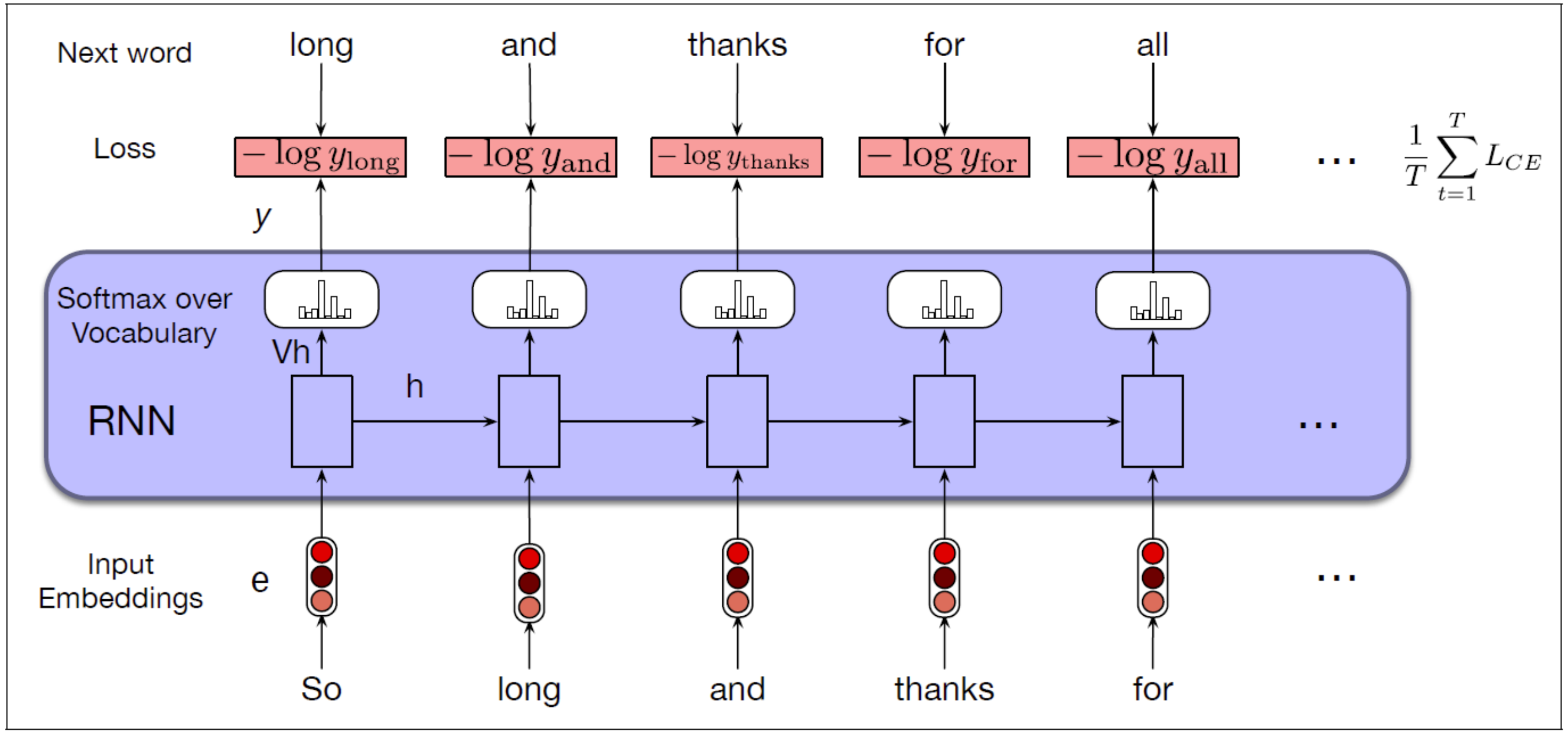


基于神经网络的自然语言处理方法

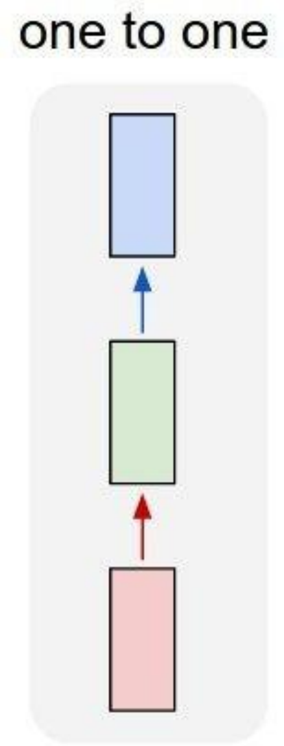
递归神经语言模型：基于前词上下文预测接下来的词

- 输入独热向量/或嵌入向量
- RNN递归生成
- 无窗口限制

$$
\begin{aligned}
\mathbf{e}_t &= \mathbf{E}\mathbf{x}_t \\
\mathbf{h}_t &= g(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{e}_t) \\
\mathbf{y}_t &= \mathrm{softmax}(\mathbf{V}\mathbf{h}_t)
\end{aligned}
$$

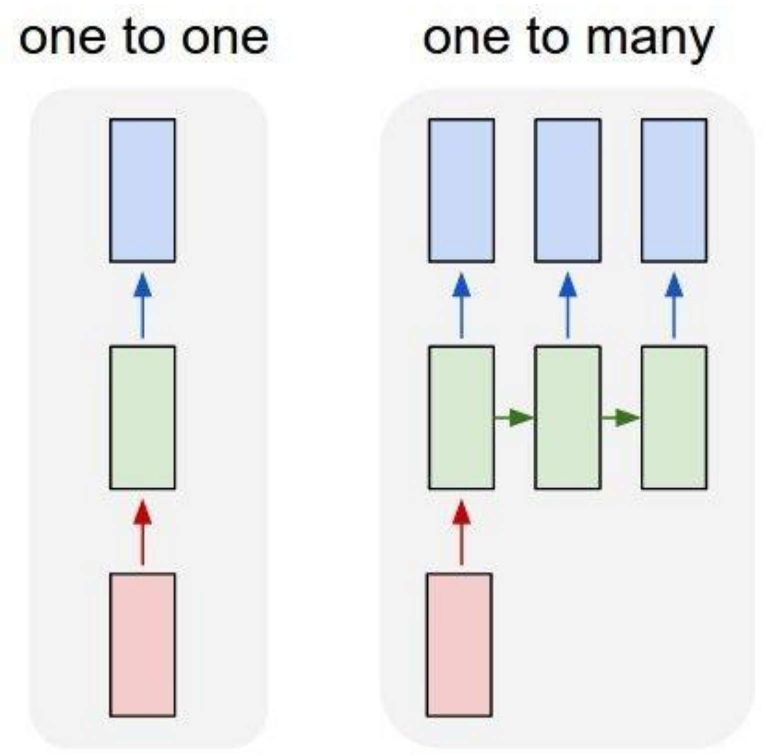


Recurrent Neural Networks



Vanilla Neural Networks

Recurrent Neural Networks



例如：图像描述

图像 -> 文字描述

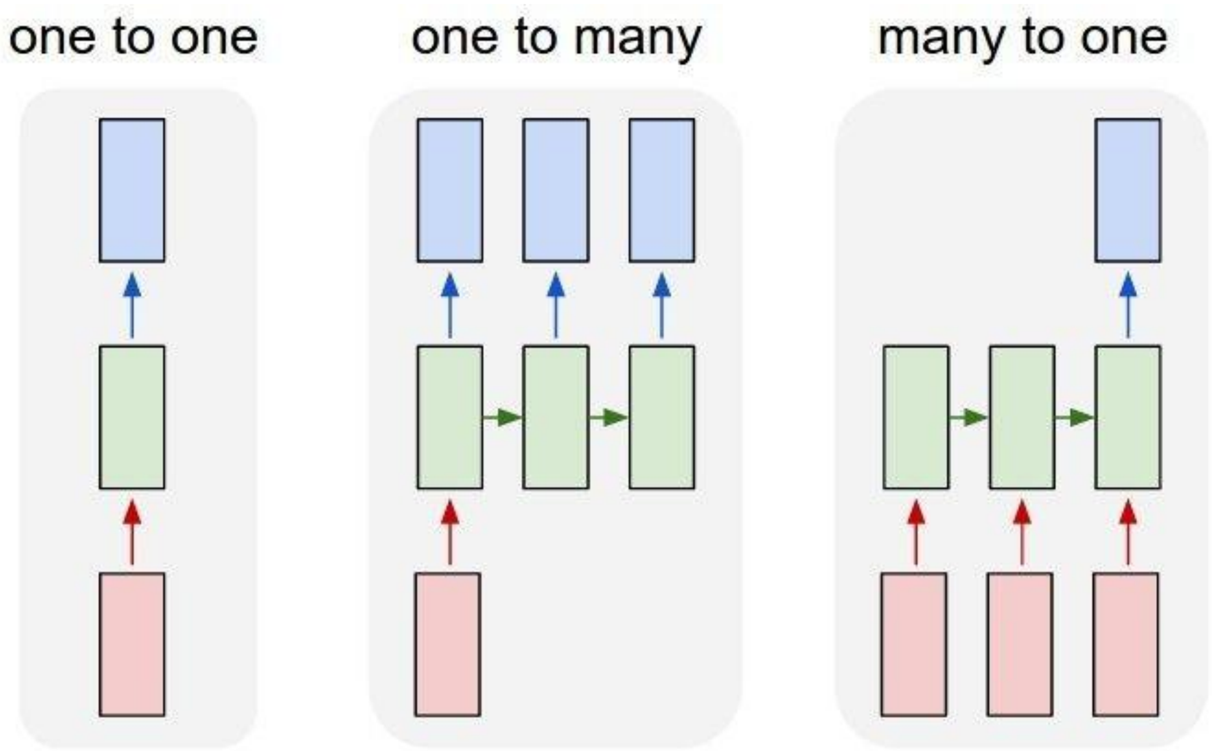
A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

Two people walking on the beach with surfboards

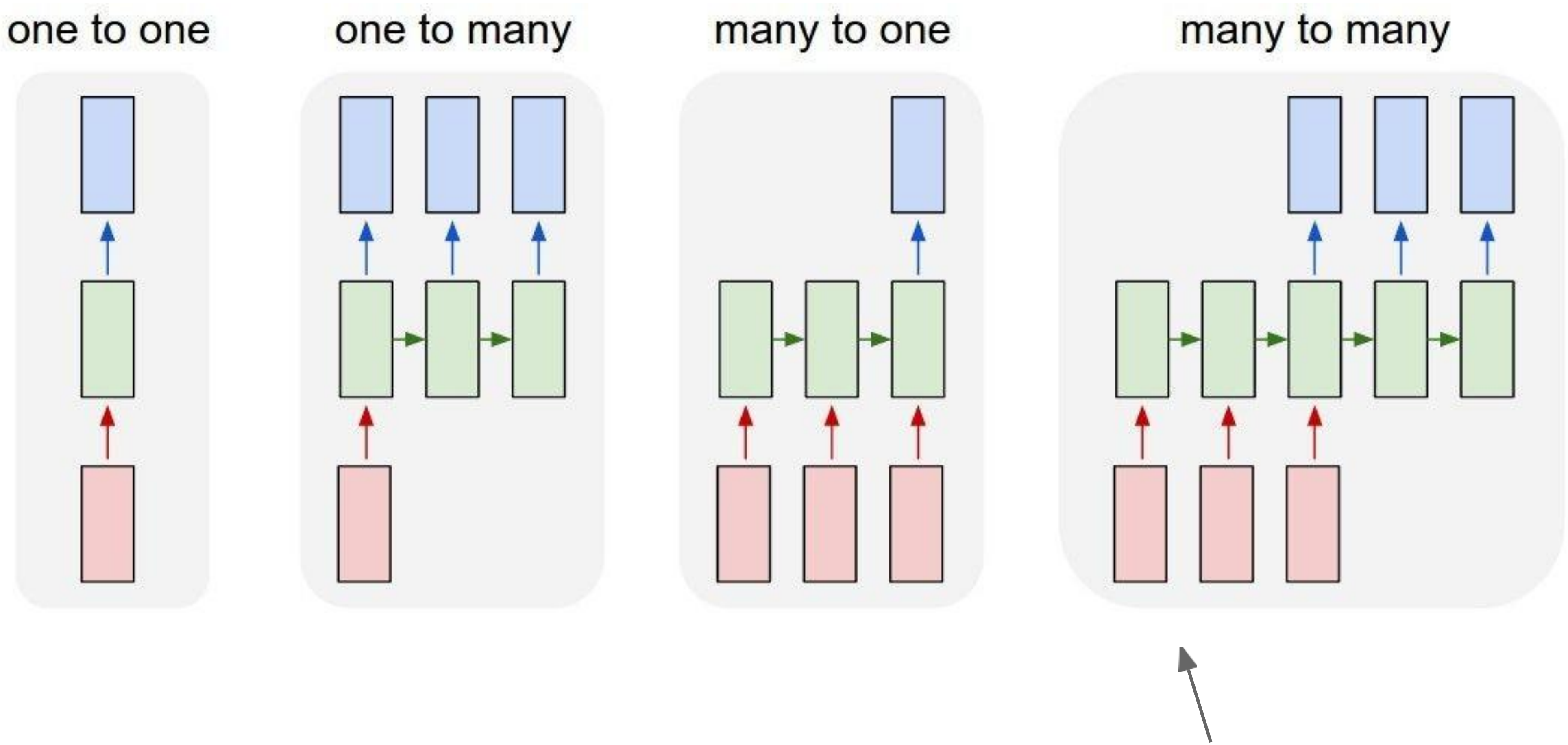
A tennis player in action on the court

Recurrent Neural Networks



动作预测
视频时间帧 -> 动作类别

Recurrent Neural Networks



例如：文本翻译

中文句子->英文句子

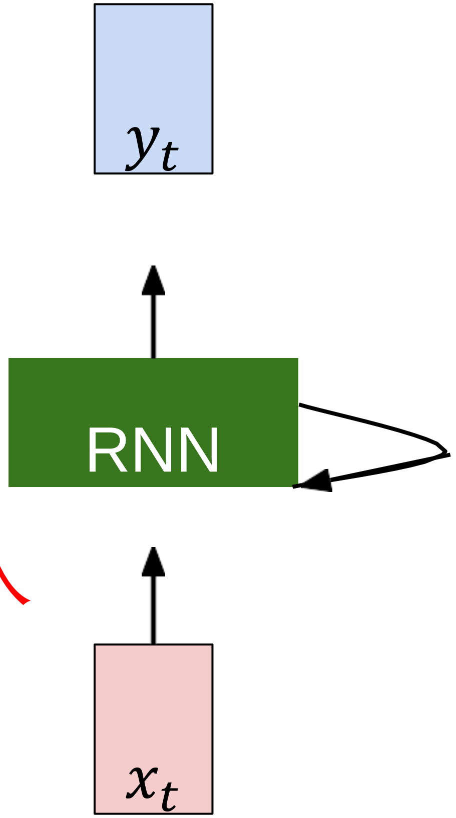
RNN隐状态更新

循环地使用RNN更新输入x每个时间步的隐状态 h_t ：

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

新状态　　旧状态　　某一时间步的输入

更新状态的函数
（参数为W）



RNN输出预测

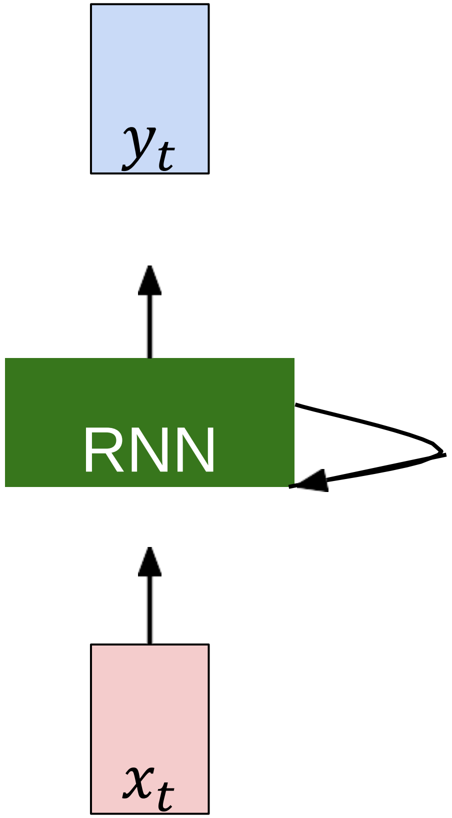
循环地使用RNN更新输入x每个时间步的预测值y_t：

$$\boxed{y_t} = \boxed{f_{W_{hy}}}(\boxed{h_t})$$

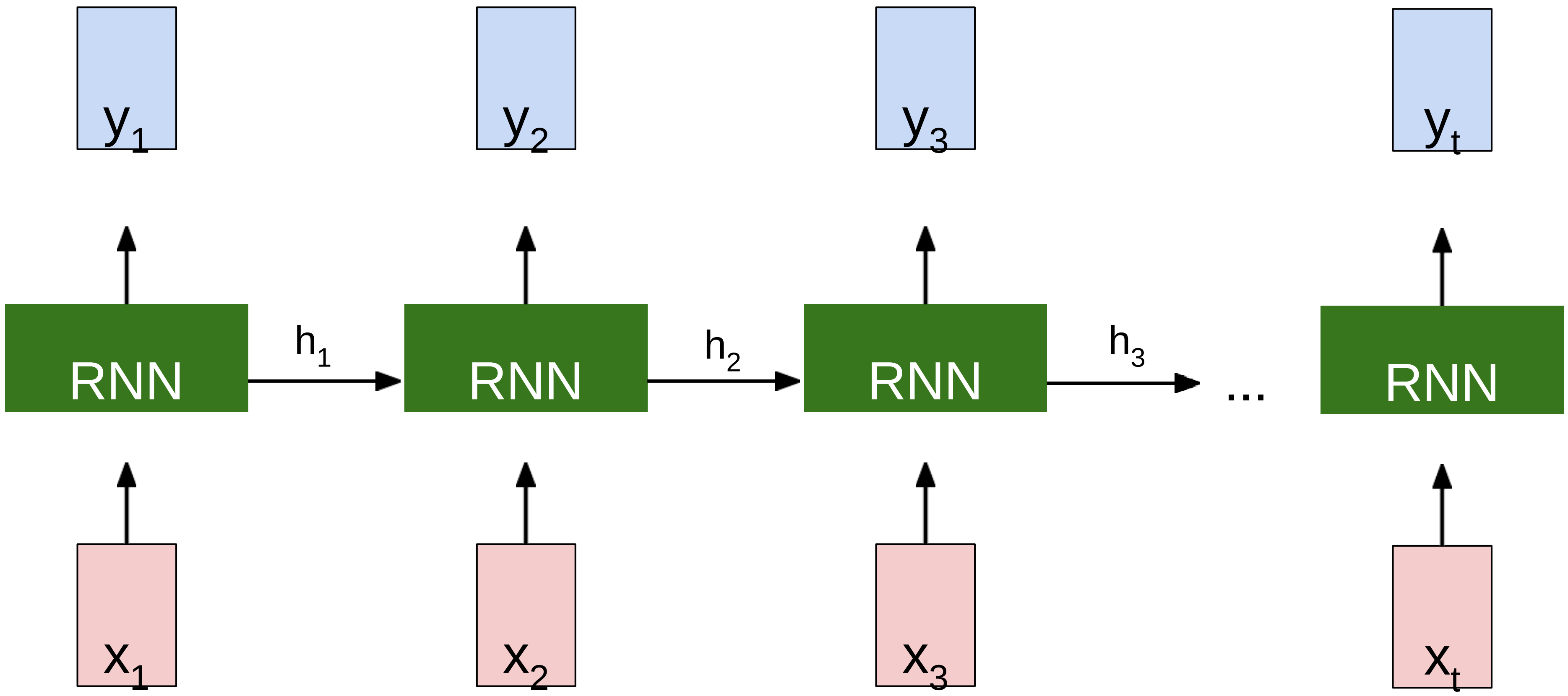
时刻t的预测

时刻t的隐状态

输出预测的函数
（参数为W_{hy}）

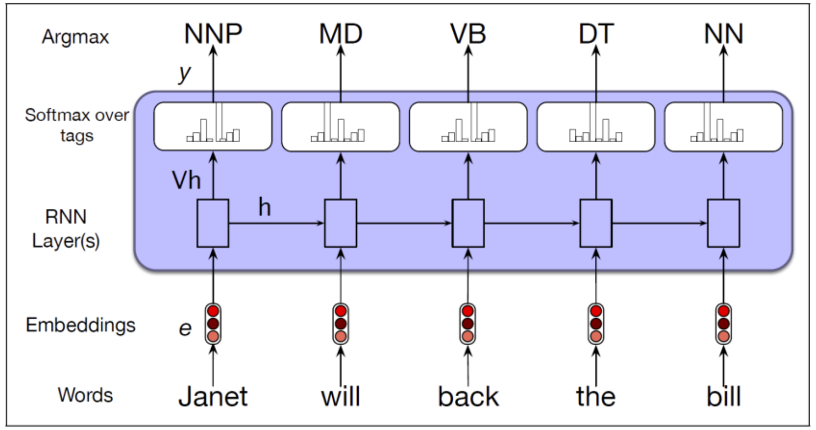


展开 RNN

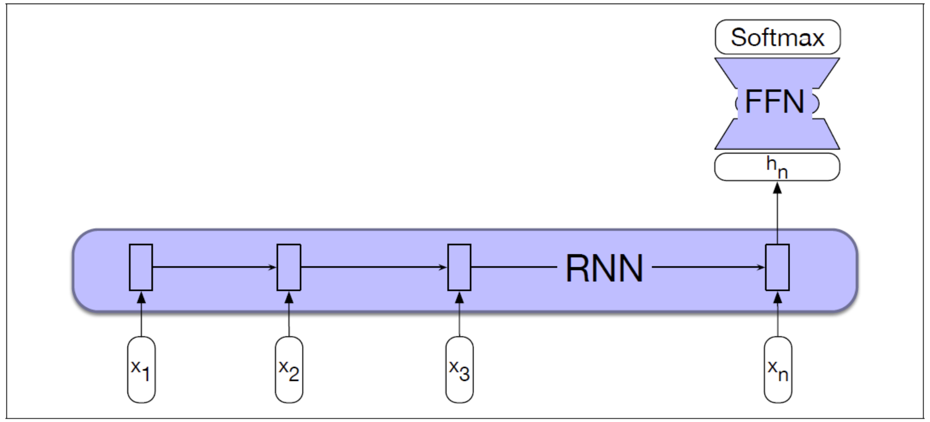


基于神经网络的自然语言处理方法

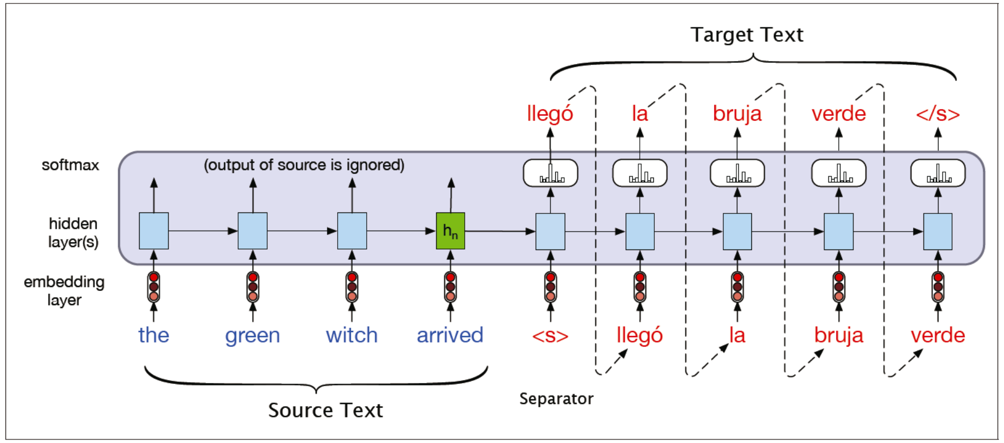
递归神经网络RNN在NLP其他任务中的应用：



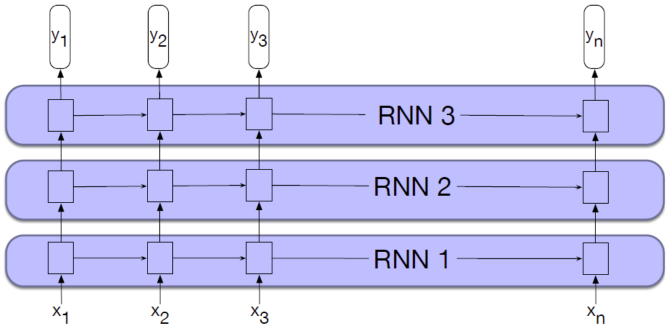
序列标注：例如词性标注



序列分类：例如情感分类、垃圾邮件分类等



文本生成：例如机器翻译、文本摘要、问答等



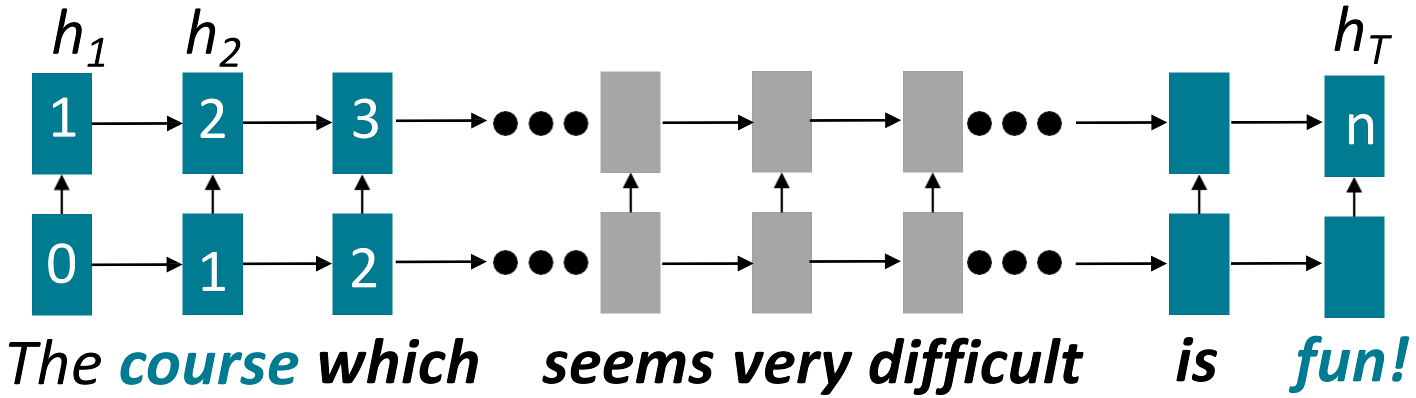
改进：多层堆叠RNN等…

RNN的优缺点

- RNN的优点:
  - 能够处理任意长度数据
  - 输入变长，模型大小不变
- RNN的缺点:
  - 未来状态的计算依赖于过去的状态（无法并行计算）
  - 很难处理长距离依赖（由于梯度衰减问题）

RNN的优缺点

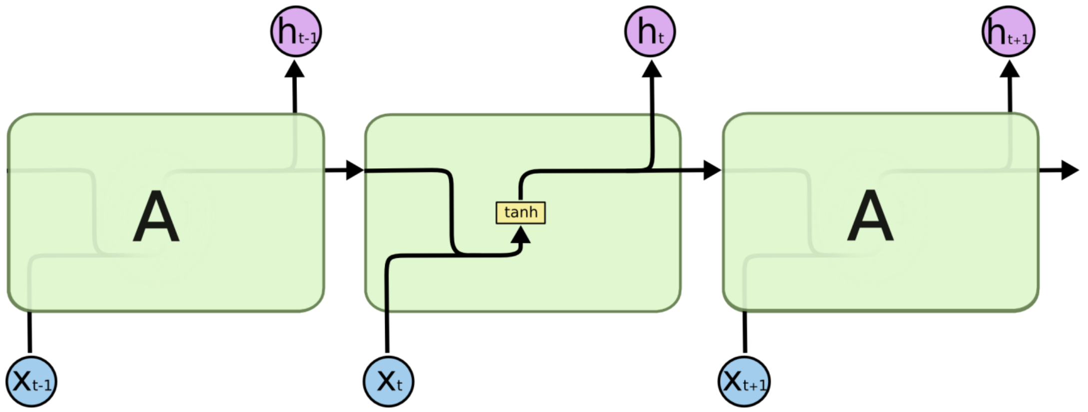
上面一列是隐状态，下面一列是输入序列。



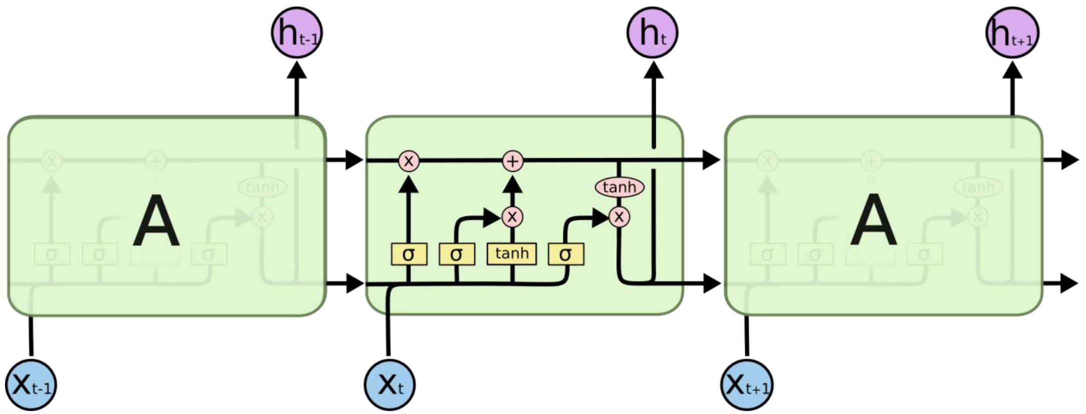
主语(course)和形容词(fun)相隔了许多词，隐状态更新太多次已经遗忘了主语。RNN很难捕获长距离的依赖！

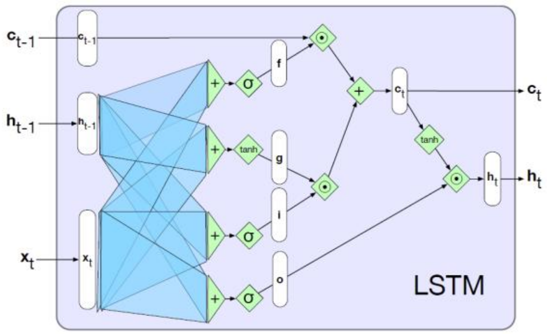
改进：LSTM

RNNs



LSTM





S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9 (8), pp. 1735–1780, 1997

谢谢

北京大學
PEKING UNIVERSITY