南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

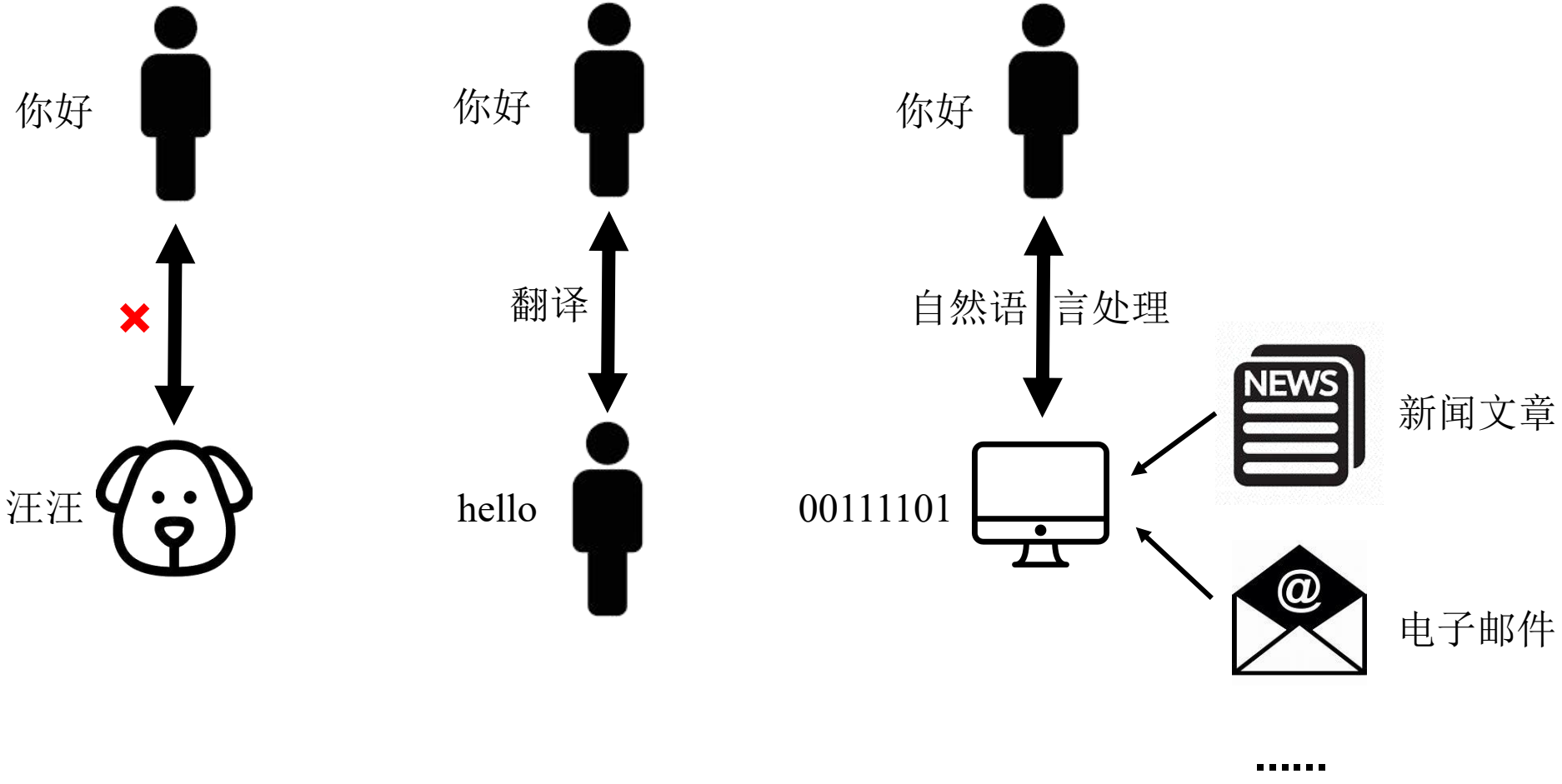
自然语言处理

魏明强、宫丽娜
计算机科学与技术学院

目录

- 自然语言处理概述
- 词嵌入
  - 独热向量
  - word2vec
    - 跳元模型
    - 连续词袋模型
- 循环神经网络（RNN）
- 长短期记忆网络（LSTM）
- 门控循环单元（GRU）

自然语言处理概述

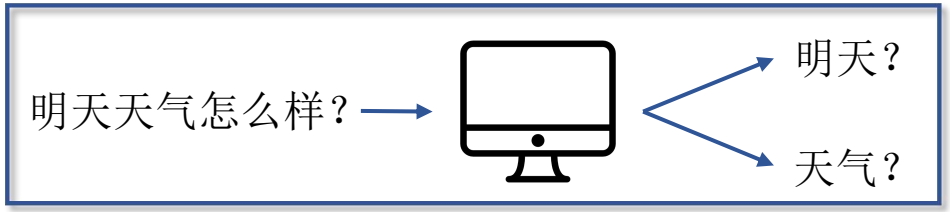


自然语言处理使计算机能够解读、处理和理解人类语言，成为人类和计算机之间沟通的桥梁

自然语言处理概述——基本任务

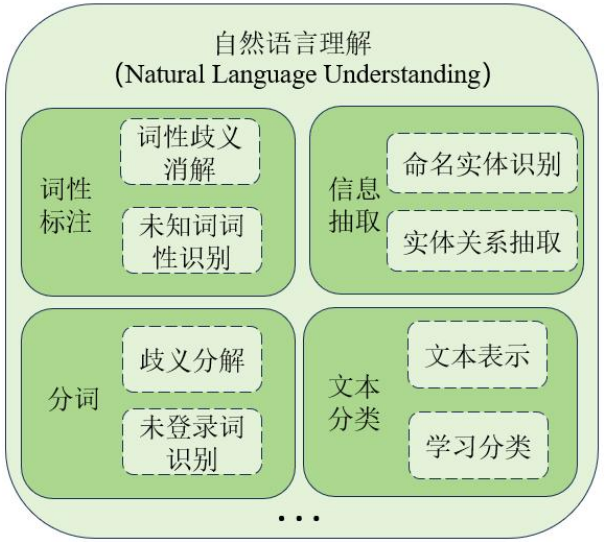
□ 自然语言理解

人与计算机交流的第一步就是让计算机理解人类输入给它的信息。这类任务的研究目的是使**计算机能够理解自然语言**，从自然语言中提取有用的信息输出或用于下游任务



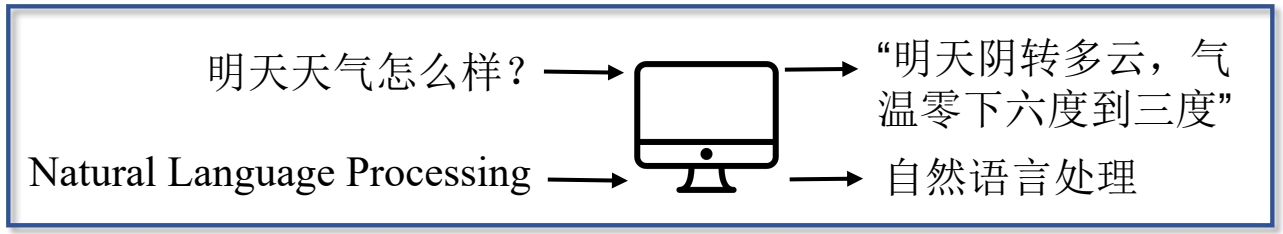
自然语言理解类任务包括：

- 词性标注
- 分词
- 文本分类
- 信息抽取



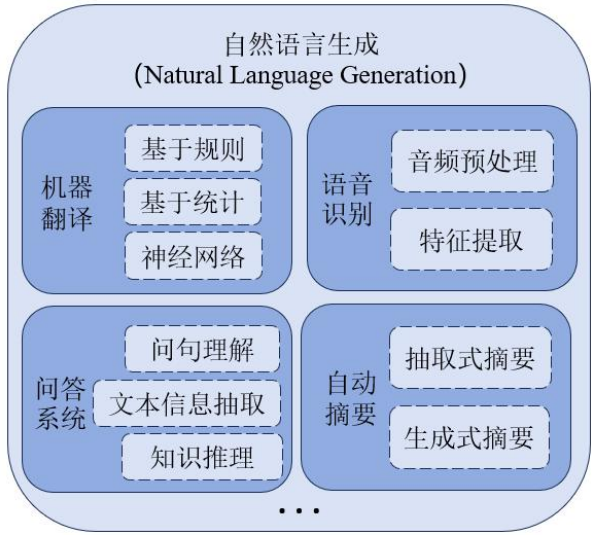
□ 自然语言生成

计算机理解人类的输入后，我们还希望计算机能够生成满足人类目的的、可以理解的**自然语言形式的输出**，从而实现真正的交流。

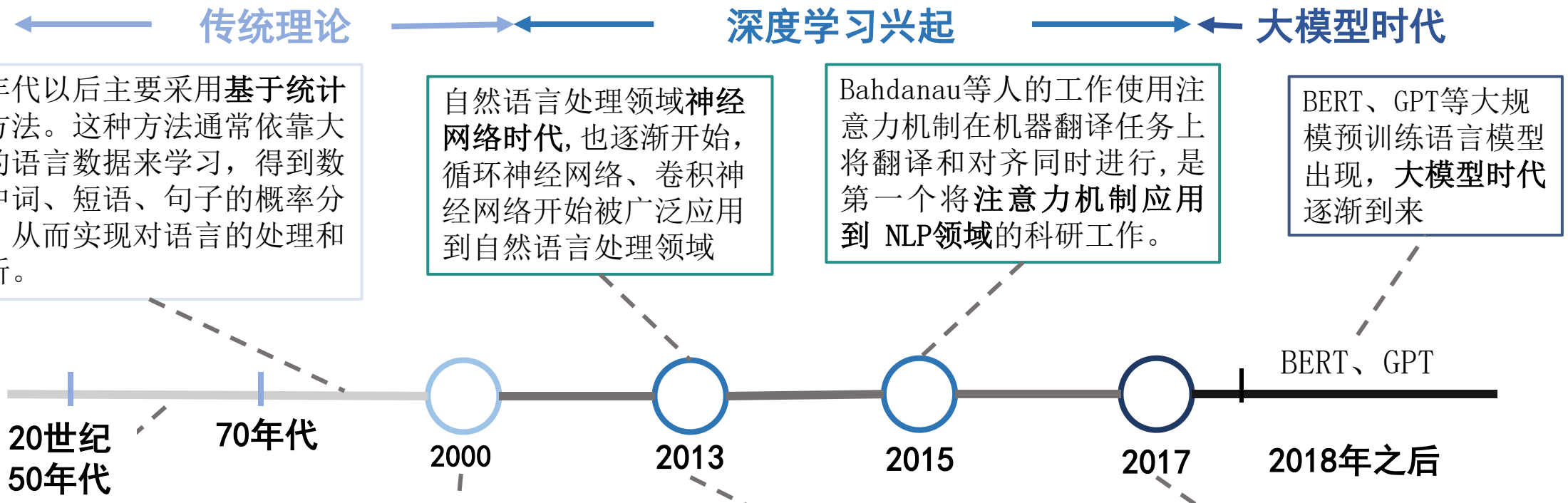


自然语言生成类任务包括：

- 机器翻译
- 问答系统
- 自动摘要
- 语音识别



自然语言处理概述——发展历程



70年代以后主要采用**基于统计**的方法。这种方法通常依靠大量的语言数据来学习，得到数据中词、短语、句子的概率分布，从而实现对语言的处理和分析。

自然语言处理领域**神经网络时代**，也逐渐开始，循环神经网络、卷积神经网络开始被广泛应用到自然语言处理领域

Bahdanau等人的工作使用注意力机制在机器翻译任务上将翻译和对齐同时进行，是第一个将**注意力机制应用到 NLP领域**的科研工作。

BERT、GPT等大规模预训练语言模型出现，**大模型时代**逐渐到来

20世纪50年代到70年代主要采用**基于规则**的方法。这种方法依赖于语言学家和开发者预先定义的规则系统，以便解析和理解语言。

Bengio等人提出**第一个神经语言模型**。这个模型将某词语之前出现的n个词语作为输入，预测下一个单词输出。模型一共三层，第一层是映射层，将n个单词映射为对应的词嵌入；第二层是隐藏层；第三层是输出层，使用softmax 输出单词的概率分布，是一个多分类器。

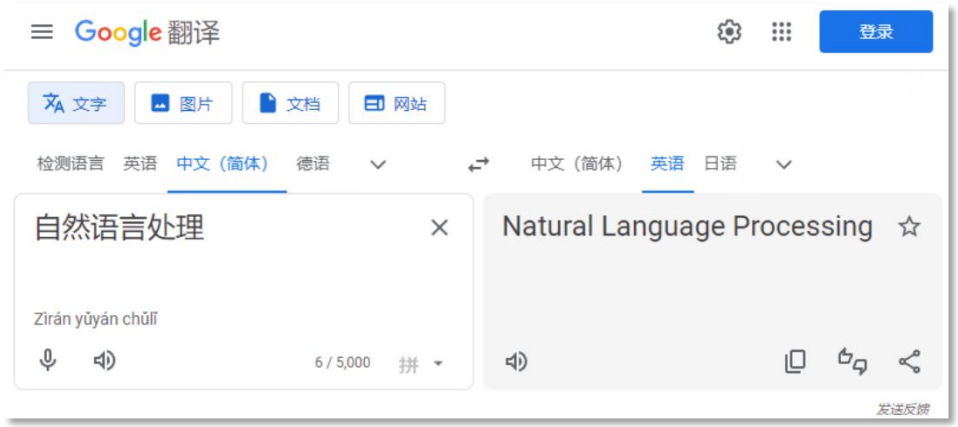
Mikolov等人提出了**word2vec**，大规模词向量的训练成为可能

Transformer提出，它创造性地用非序列模型来处理序列化的数据，并且大获成功。

自然语言处理概述——应用领域

南京航空航天大学 NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

1. 翻译软件



2. 聊天机器人

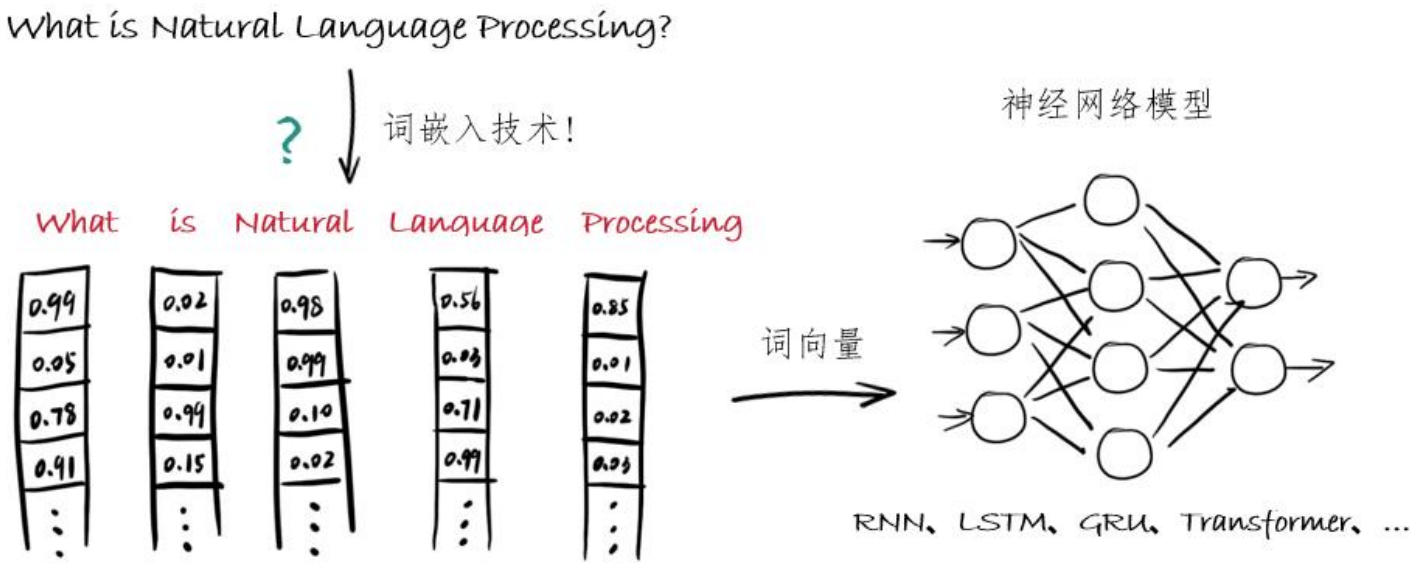
3. 语音助手

4. 搜索引擎

目录

? 计算机是无法直接读懂非数值的自然语言，
只有将其**转化为数值形式**才能被计算机处理 → 词嵌入

? 完成各种下游任务 → 神经网络模型 → 循环神经网络（RNN）
→ 长短期记忆网络（LSTM）
→ 门控循环单元（GRU）



- 自然语言处理概述
- 词嵌入
  - 独热向量
  - word2vec
    - 跳元模型
    - 连续词袋模型
- 循环神经网络（RNN）
- 长短期记忆网络（LSTM）
- 门控循环单元（GRU）

词嵌入——独热向量(One-hot Encoding)

文本 $\xrightarrow{?}$ 数值　最简单的方法就是用独热向量表示每个单词

- 独热向量是指使用 N 位 0 或 1 对 N 个单词进行编码，其分量和类别数一样多，**类别对应的分量设置为1**（即 one-hot），其余分量设置为 0。

例如，编码apple、bag、cat、dog、elephant五个单词，用 5 位向量进行编码：

apple = [**1** 0 0 0 0]

bag = [0 **1** 0 0 0]

cat = [0 0 **1** 0 0]

dog = [0 0 0 **1** 0]

elephant = [0 0 0 0 **1**]

$$cos(x,y) = \frac{x^{\top}y}{\|x\|\|y\|} = 0$$

但任意两词之间余弦相似度为0！

独热向量的维度等于词汇表大小，在词汇表较大时会变得非常长

√ **优点：**

- 独热向量容易构建

× **缺点：**

- 独热向量不能编码词之间的相似性
- 特征矩阵非常稀疏，占用空间很大

词嵌入——word2vec

我们希望词向量：

- **携带上下文信息**，即词与词之间的联系能在词的向量表示中体现。
- 词的表示是稠密的，能用**更少的空间、更低的维数**表示更多的信息。

我们希望实现这样的效果：

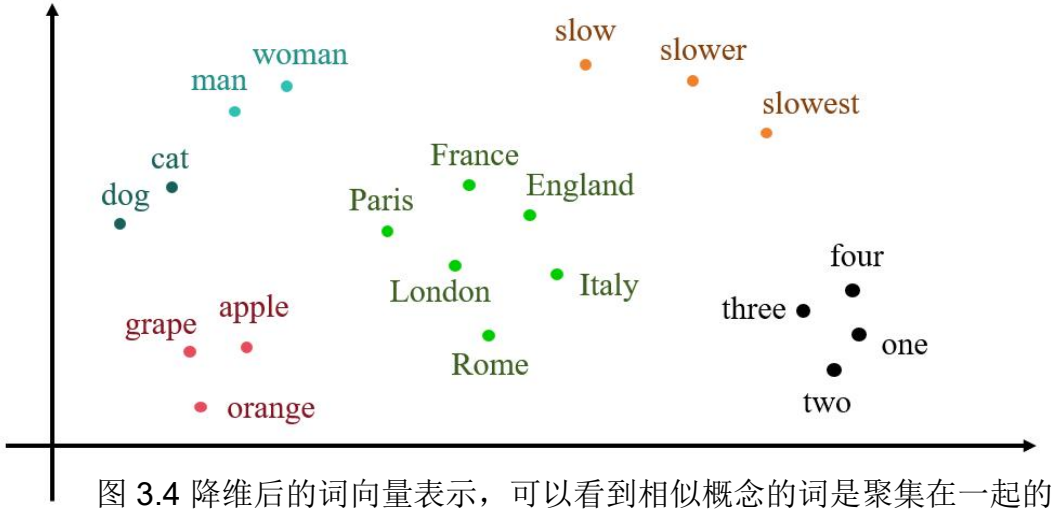


图 3.4 降维后的词向量表示，可以看到相似概念的词是聚集在一起的

实现 ← **word2vec!**

- word2vec是一种词嵌入技术，也可被看作是一个神经网络模型，其参数是词向量，通过**预测上下文**来学习好的词向量。

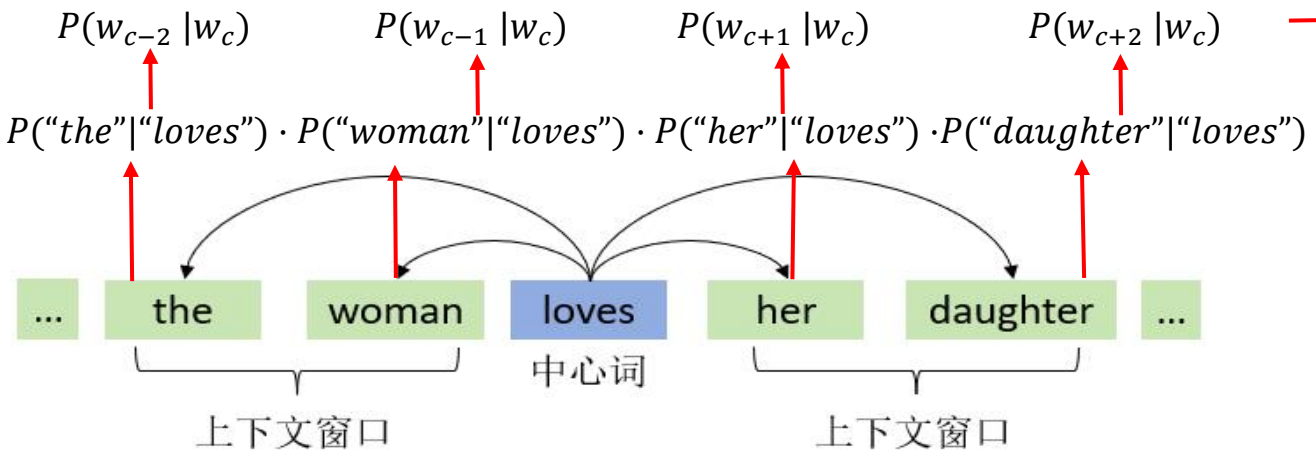
和独热向量相比，word2vec生成的词向量具有以下优点：

- 训练时利用上下文信息，词向量包含词的语义信息和词与词之间的联系。
- 维度更少，所以占用空间更少、计算成本更低。
- 通用性强，可用于各种下游 NLP 任务。

训练word2vec的常用方法有两种：跳元模型（Skip-Gram）和连续词袋（Continuous Bagsof Words：CBOW）

词嵌入——跳元模型

- 根据**中心词**预测上下文词

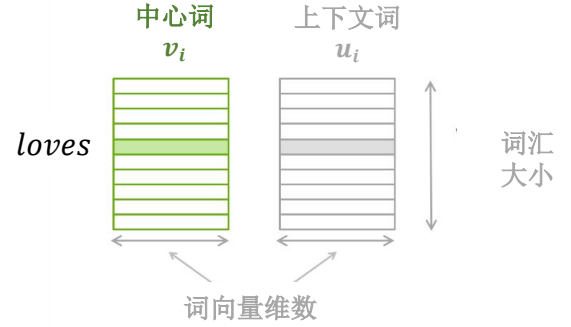


如何计算？

$$P(w_o|w_c) = \frac{exp(u_o{}^T v_c)}{\sum_{i \in V} exp(u_i{}^T v_c)}$$ 就是 *softmax*！

对于每个单词w_i我们有两个向量：

- 当它是一个中心词时，用$\boldsymbol{v_i}$表示该词
- 当它是一个上下文词时，用$\boldsymbol{u_i}$表示该词



- 目标函数（损失函数）
  - 给定长度为 T 的文本序列，时刻 t 处的词表示为 $w^{(t)}$
  - 上下文窗口大小为 m

则，似然函数为在给定任何中心词的情况下生成所有上下文的概率：

$$Likehood = \prod_{t=1}^{T} \prod_{-m \le j \le m, j \ne 0} P(w^{(t+j)}|w^{(t)})$$

- 目标是最大化该似然函数，即最小化损失函数：

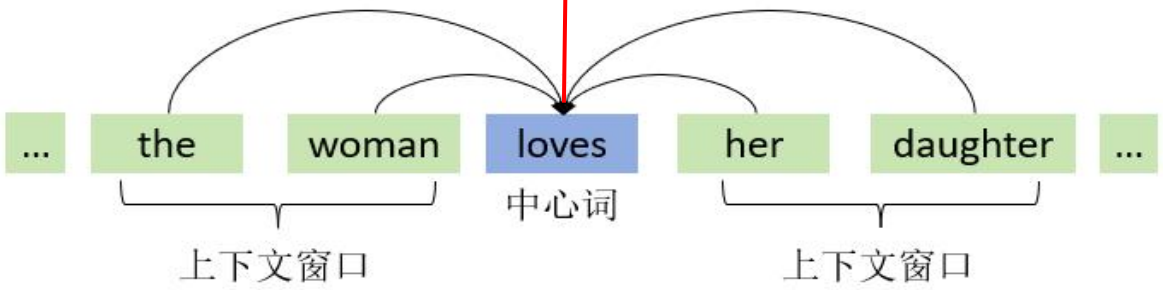
$$Loss = -\sum_{t=1}^{T} \sum_{-m \le j \le m, j \ne 0} P(w^{(t+j)}|w^{(t)})$$

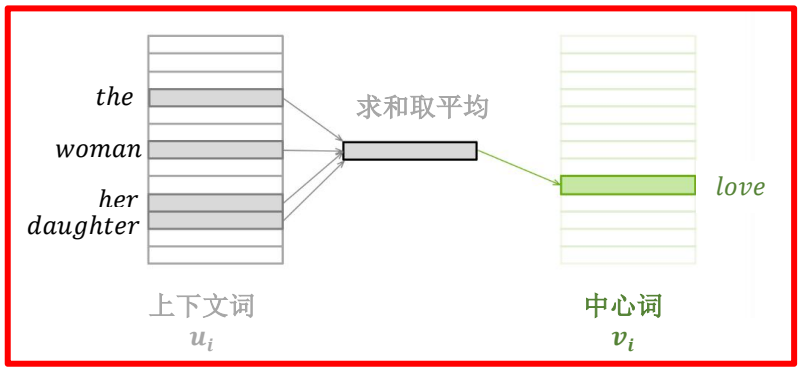
词嵌入——连续词袋模型

- 根据上下文词预测中心词

$P(w_c \mid w_{c-2}, w_{c-1}, w_{c+1}, w_{c+2})$ 如何计算？ $P(w_c \mid w_{o_1}, \ldots, w_{o_{2m}}) = \frac{exp(\frac{1}{2m} u_c{}^T (v_{o_1} + \ldots + v_{o_{2m}}))}{\sum_{i \in V} exp(\frac{1}{2m} u_i{}^T (v_{o_1} + \ldots + v_{o_{2m}}))}$

$P(\text{“loves”} \mid \text{“the”}, \text{“woman”}, \text{“her”}, \text{“daughter”})$





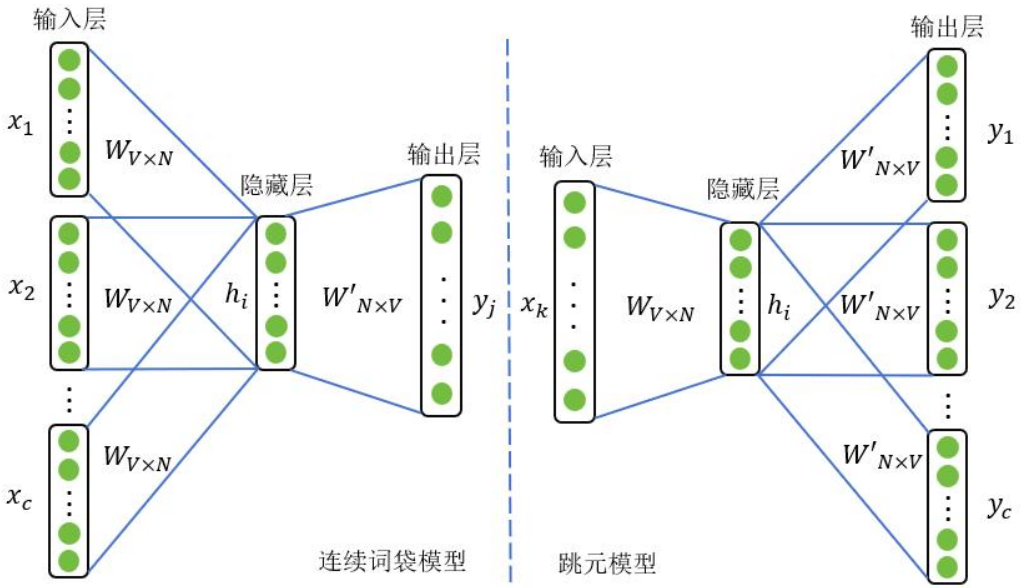
- 目标函数（损失函数）

- 给定长度为 T 的文本序列，时刻 t 处的词表示为 $w^{(t)}$
- 上下文窗口大小为 m

则，似然函数为在给定上下文词的情况下生成所有中心词的概率：$Likehood = \prod_{t=1}^{T} P(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$

- 目标是最大化该似然函数，即最小化损失函数：$Loss = -\sum_{t=1}^{T} P(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$

词嵌入——连续词袋模型举例



使用单词的独热编码作为输入：

the = [1 0 0 0 0]

woman = [0 1 0 0 0]

loves = [0 0 1 0 0]

her = [0 0 0 1 0]

daughter = [0 0 0 0 1]

假设训练得到的权重矩阵$W_{V\times N}$为：$\begin{bmatrix} 1 & 2 & 3 & 0 & 1 \\ 1 & 2 & 1 & 2 & 2 \\ -1 & 1 & 1 & 1 & 0 \end{bmatrix}$

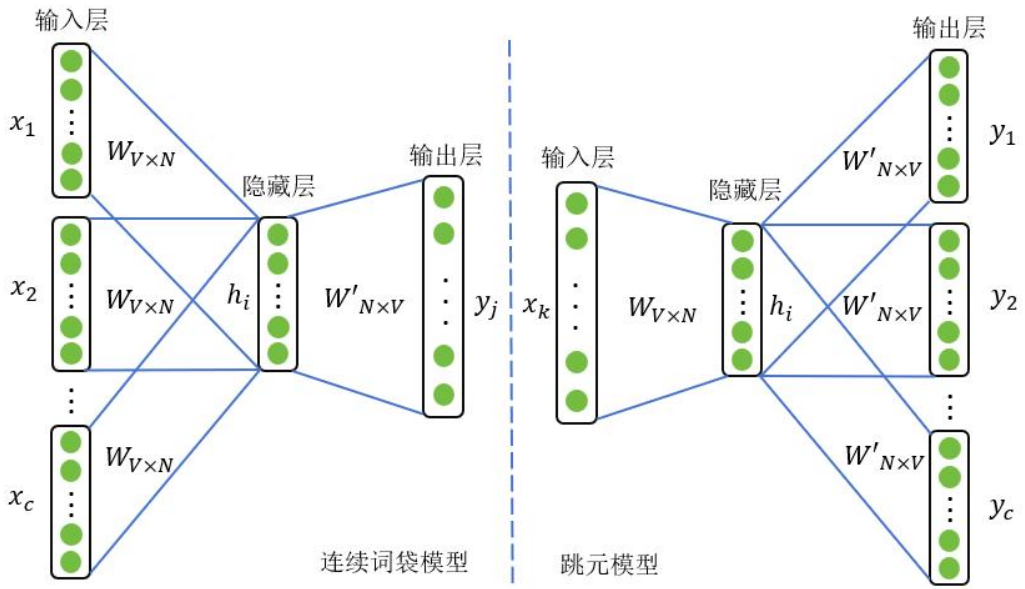
其中，N=5 表示输入层单词的维数，V=3 表示希望得到的词向量维数

现在将 “the” 输入，即与权重矩阵相乘：$\begin{bmatrix} 1 & 2 & 3 & 0 & 1 \\ 1 & 2 & 1 & 2 & 2 \\ -1 & 1 & 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$

“the”的词向量

同理，可以得到每个单词的词向量为：$woman = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}, her = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}, daughter = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$

词嵌入——连续词袋模型举例



将得到的4个向量相加求平均作为输出层的输入：

$$\frac{1}{4}\left(\begin{bmatrix}1\\1\\-1\end{bmatrix}+\begin{bmatrix}2\\2\\1\end{bmatrix}+\begin{bmatrix}0\\2\\1\end{bmatrix}+\begin{bmatrix}1\\2\\0\end{bmatrix}\right)=\begin{bmatrix}1.00\\1.75\\0.25\end{bmatrix}$$

将该向量与权重矩阵$W'_{N\times V}$（也是网络训练的结果）相乘得到输出向量，最后将 softmax 作用在输出向量上，得到每个词的概率分布:

$$softmax\left(\begin{bmatrix}1&2&-1\\-1&2&-1\\1&2&2\\0&2&0\\1&-1&2\end{bmatrix}\begin{bmatrix}1.00\\1.75\\0.25\end{bmatrix}\right)=softmax\left(\begin{bmatrix}4.250\\2.250\\5.000\\3.000\\-0.250\end{bmatrix}\right)=\begin{bmatrix}0.268\\0.036\\0.567\\0.126\\0.003\end{bmatrix}$$

最后计算损失函数，反向传播，更新网络参数。

！预测目标单词是训练网络的方式，获得网络的中间产物——权重矩阵$W_{V\times N}$才是期望得到的，因为任意一个单词的独热向量乘该矩阵就能得到自己的词向量

目录

- 自然语言处理概述
- 词嵌入
  - 独热向量
  - word2vec
    - 跳元模型
    - 连续词袋模型
- 循环神经网络（RNN）
- 长短期记忆网络（LSTM）
- 门控循环单元（GRU）

循环神经网络（RNN）

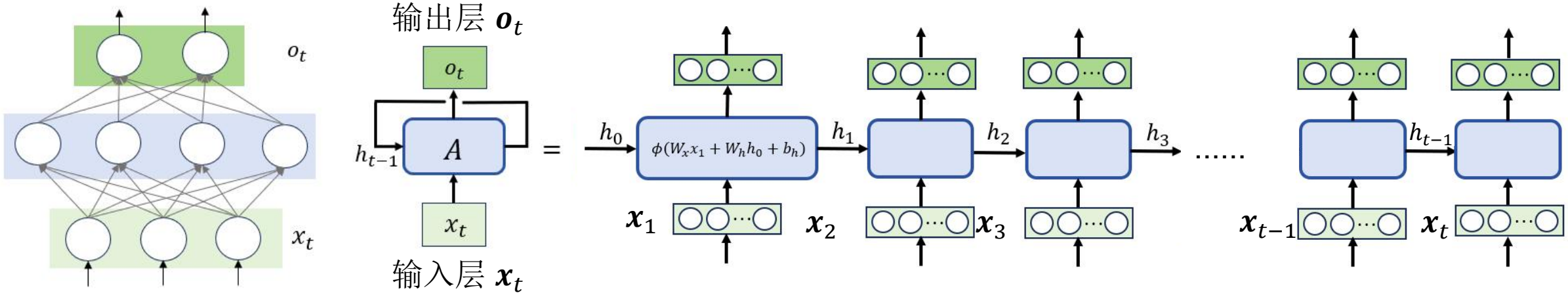
时序的重要性！

✖ working love learning we on deep

√ we love working on deep learning

捕捉序列中的时序信息 ⟵ 循环神经网络（RNN）

□ 循环神经网络



当前时刻的隐藏状态 $\boldsymbol{h_t}$：$\boldsymbol{h_t = \phi(W_x x_t + W_h h_{t-1} + b_n)}$ ⟶ 有效包含当前输入和先前序列的信息

当前时刻的输出 $\boldsymbol{o_t}$：$\boldsymbol{o_t = W_o h_t + b_o}$

循环神经网络——训练

- BPTT (Back Propagation Through Time) 算法

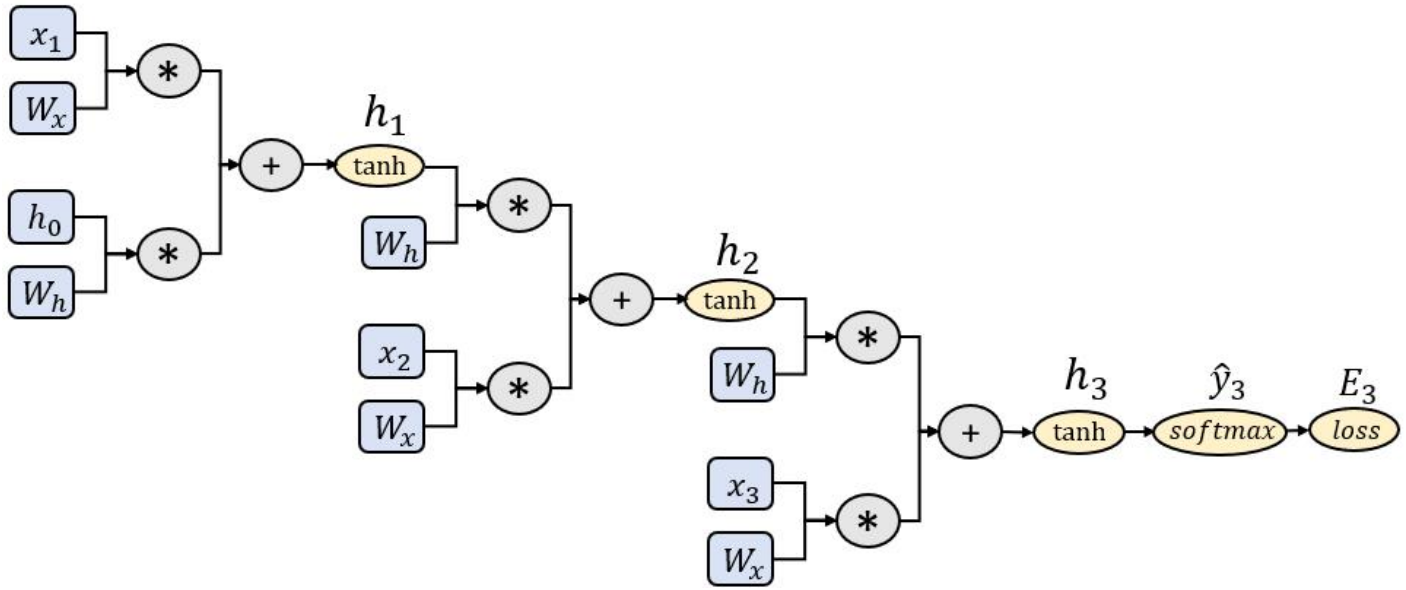
假设当前时刻的隐藏状态和输出为：

$$\boldsymbol{h}_t = \phi(\boldsymbol{W}_x\boldsymbol{x}_t + \boldsymbol{W}_h\boldsymbol{h}_{t-1})$$

$$\hat{\boldsymbol{y}}_t = softmax(\boldsymbol{W}_o\boldsymbol{h}_t)$$

则可以使用交叉熵计算每个时刻的损失，则在 t=3 时有损失：

$$E_3 = -\boldsymbol{y}_3 log\hat{\boldsymbol{y}}_3$$



以t=3时刻的损失E_3为例，对参数$\boldsymbol{W}_o$的梯度为：

$$\frac{\partial E_3}{\partial \boldsymbol{W}_o} = \frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{W}_o}$$

损失E_3对参数$\boldsymbol{W}_x$的梯度：$\frac{\partial E_3}{\partial \boldsymbol{W}_x} = \frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{h}_3}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{W}_x} + \frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{h}_3}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{W}_x} + \frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{h}_3}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{h}_1}\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_x}$

损失E_3对参数$\boldsymbol{W}_h$的梯度：$\frac{\partial E_3}{\partial \boldsymbol{W}_h} = \frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{h}_3}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{W}_h} + \frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{h}_3}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{W}_h} + \frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{h}_3}\frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{h}_1}\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_h}$

简化表达

$$\frac{\partial E_3}{\partial \boldsymbol{W}_x} = \sum_{k=1}^{3}\frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial h_3}\left(\prod_{j=k+1}^{3}\frac{\partial \boldsymbol{h}_j}{\partial \boldsymbol{h}_{j-1}}\right)\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{W}_x}$$

$$\frac{\partial E_3}{\partial \boldsymbol{W}_h} = \sum_{k=1}^{3}\frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{h}_3}\left(\prod_{j=k+1}^{3}\frac{\partial \boldsymbol{h}_j}{\partial \boldsymbol{h}_{j-1}}\right)\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{W}_h}$$

循环神经网络——梯度问题

- 本质上都是因为梯度反向传播中的**连乘效应**，**小于1的数**连乘就会出现**梯度下降**问题，**大于1的数**连乘就会出现**梯度爆炸**的问题

- **梯度消失问题**

在E_3对参数$\boldsymbol{W}_x$和$\boldsymbol{W}_h$的偏导数的公式中存在连乘部分：$\frac{\partial E_3}{\partial \boldsymbol{W}_x}=\sum_{k=1}^{3}\frac{\partial E_3}{\partial \hat{\boldsymbol{y}}_3}\frac{\partial \hat{\boldsymbol{y}}_3}{\partial \boldsymbol{h}_x}\left(\prod_{j=k+1}^{3}\frac{\partial \boldsymbol{h}_j}{\partial \boldsymbol{h}_{j-1}}\right)\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{W}_x}$

假设当激活函数为Tanh时，连乘部分可以表示为：$\prod_{j=k+1}^{3}\frac{\partial \boldsymbol{h}_j}{\partial \boldsymbol{h}_{j-1}}=\prod_{j=k+1}^{3}Tanh'\boldsymbol{W}_h$

而Tanh的导数可以写为：$Tanh'(x)=(1-Tanh^2(x))$，其值域为：$(0,1]$

当Tanh'与$\boldsymbol{W}_h$乘积小于1时，假设有n项连乘：$(\text{小于1的数})^n$。随着时刻向前推移，梯度呈指数级下降。

- **梯度爆炸问题**

那么如果Tanh'与$\boldsymbol{W}_h$的乘积大于1，假设有n项连乘，则可以表示为$(\text{大于1的数})^n$。随着时刻向前推移，梯度是呈指数级上升的，也就是梯度爆炸的问题。

循环神经网络——梯度问题的缓解

无论是循环神经网络还是卷积神经网络等其他网络，它们出现梯度消失和梯度爆炸问题的本质原因是一样的，因此对于这些网络来说存在一些共性的缓解方案：

- **更换激活函数**：根据前面的分析，当使用Tanh激活函数时，若Tanh'值过小会导致梯度消失问题，此时可以更换为 Relu 激活函数。根据第二章介绍可以得到，Relu 函数的导数在正数部分恒为 1，因此解决了由于激活函数导数的值过小导致的梯度消失问题，起到了缓解作用。
- **使用批归一化**：正如第二章所述，其思想是对每一层的输入进行归一化，使其均值接近0，标准差接近 1。这样，输入值就能落在激活函数的梯度非饱和区，也就是梯度较大的区域，从而缓解梯度消失问题。
- **梯度裁剪**：该方案主要针对梯度爆炸问题，其思想是对梯度设置一个裁剪阈值，在更新梯度时如果梯度超过这个阈值则将其设置为阈值范围内，因此能够缓解梯度爆炸的情况。

循环神经网络——双向RNN

我很高兴

我__不__困，我刚起床

我__非常__困，我想赶紧睡觉

□ 短语的“下文”在填空任务中起到十分关键的作用，它传达的信息关乎到选择什么词来填空。如果无法利用这一特性，普通的 RNN 模型将在相关任务上表现不佳。而既可以学习正向特征也可以学习反向特征的双向循环神经网络在解决该类任务时会有更高的拟合度。

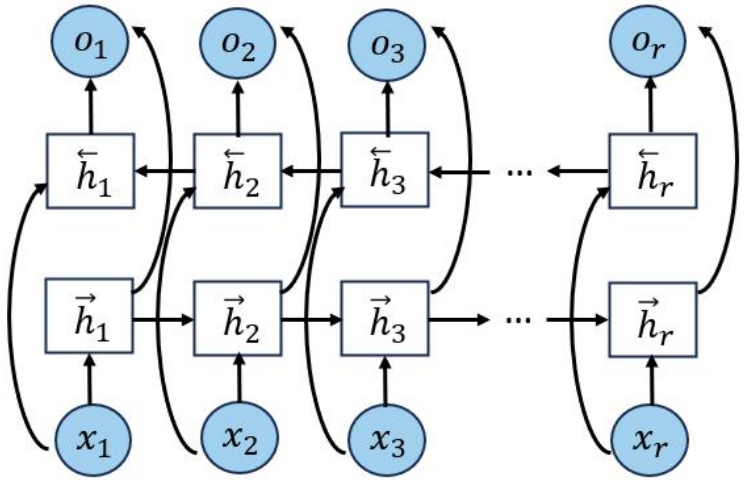


图 3.10 双向循环神经网络架构

前向隐藏状态 $\overrightarrow{\boldsymbol{h}}_t$：$\overrightarrow{\boldsymbol{h}}_t = \phi(\boldsymbol{W}_x^{(f)}\boldsymbol{x}_t + \boldsymbol{W}_h^{(f)}\overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_h^{(f)})$

反向隐藏状态 $\overleftarrow{\boldsymbol{h}}_t$：$\overleftarrow{\boldsymbol{h}}_t = \phi(\boldsymbol{W}_x^{(b)}\boldsymbol{x}_t + \boldsymbol{W}_h^{(b)}\overleftarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_h^{(b)})$

将前向和反向隐藏状态连接起来，获得用于输出层的隐藏状态：$\boldsymbol{h}_t$

输出为：$\boldsymbol{o}_t = \boldsymbol{W}_o\boldsymbol{h}_t + \boldsymbol{b}_o$

目录

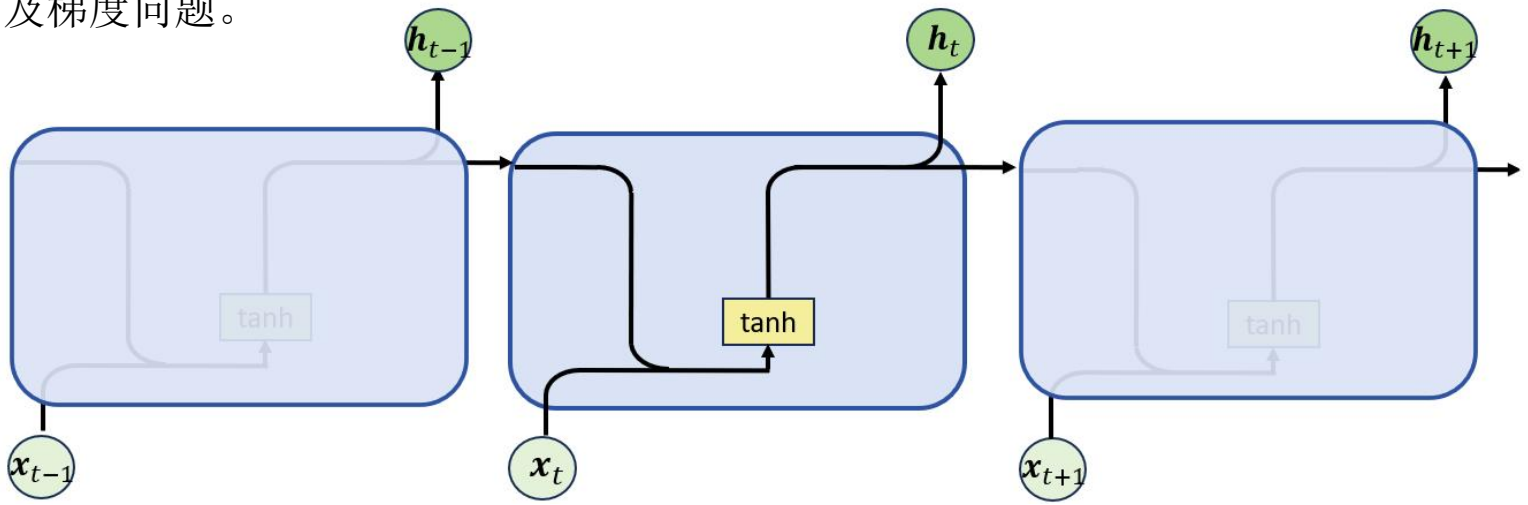
- 自然语言处理概述
- 词嵌入
  - 独热向量
  - word2vec
    - 跳元模型
    - 连续词袋模型
- 循环神经网络（RNN）
- 长短期记忆网络（LSTM）
- 门控循环单元（GRU）

长短期记忆网络(LSTM)

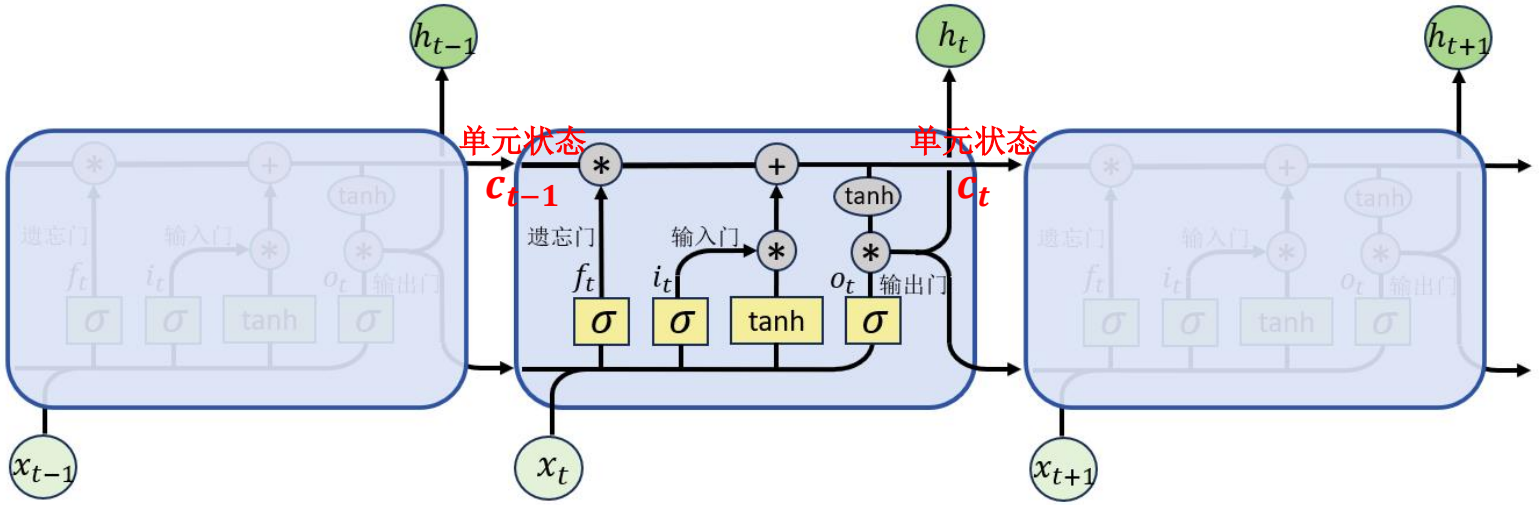
南京航空航天大学 NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

当训练深层网络时，RNN面临梯度在反向传播过程中消失或爆炸的问题。而由于梯度消失的问题，普通RNN**难以学习和记忆过去很长时间里的输入信息**，这个问题在处理长序列和复杂序列模式时变得尤为明显。长短期记忆网络的出现缓解了长期信息保存以及梯度问题。

普通RNN



LSTM

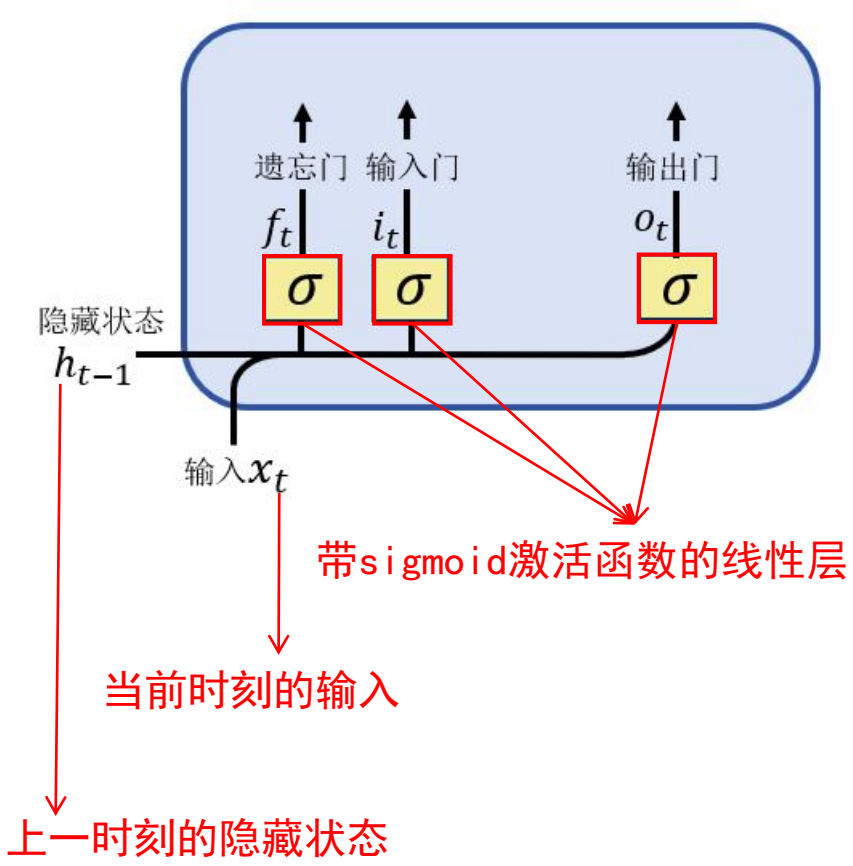


和普通RNN比较，LSTM主要是改变了隐藏层的结构。

LSTM引入了**记忆元**（memory cell）的概念，简称**单元**（cell），其设计目的是用于**记录附加信息**。

引入了**门机制**对当前的输入信息进行筛选，从而决定**哪些信息可以传递到下一层**

长短期记忆网络(LSTM)



1. 遗忘门、输入门和输出门

遗忘门：$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_x^{(f)}\boldsymbol{x}_t + \boldsymbol{W}_h^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)})$

输入门：$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_x^{(i)}\boldsymbol{x}_t + \boldsymbol{W}_h^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)})$

输出门：$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_x^{(o)}\boldsymbol{x}_t + \boldsymbol{W}_h^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)})$

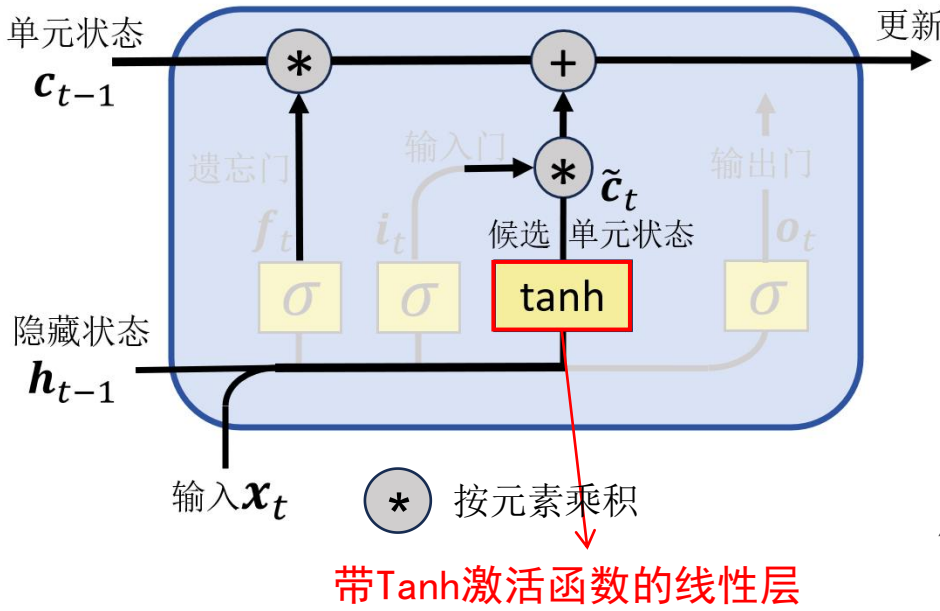
门是一种**选择性地让信息通过**的方式。σ(sigmoid)使输出保持在区间(0, 1)，越接近0表示‘不让任何内容通过’，越接近1则表示‘让所有内容通过’

遗忘门：决定模型会从单元状态中丢弃什么信息。

输入门：决定模型要从候选单元状态中保存什么信息。

输出门：决定模型要将单元状态中的什么信息传递给隐藏状态。

长短期记忆网络(LSTM)



2. 候选单元状态

$$\tilde{c}_t = Tanh(W_x^{(c)} x_t + W_h^{(c)} h_{t-1} + b^{(c)})$$

3. 单元状态更新

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

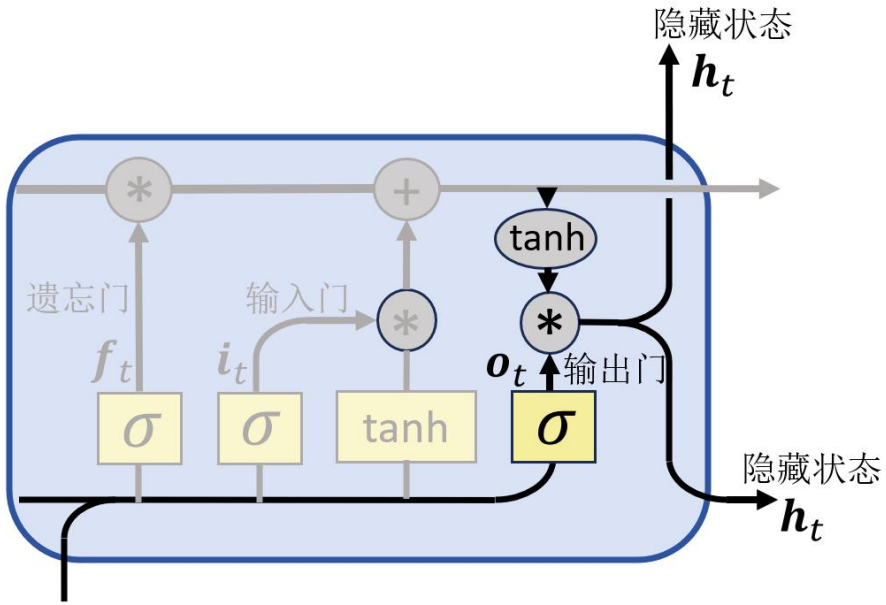
两个门f_t和i_t控制在单元状态中存储哪些新信息

遗忘门f_t：决定模型会从单元状态中丢弃什么信息

输入门i_t：决定模型要从候选单元状态中保存什么信息

例如，若遗忘门f_t接近 1 且输入门i_t接近 0，则上一时刻的单元状态c_{t-1}将随时间被保存，并传递到当前时刻。

长短期记忆网络(LSTM)



4. 隐藏状态更新

$$o_t = \sigma(W_x^{(o)} x_t + W_h^{(o)} h_{t-1} + b^{(o)})$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

$$h_t = o_t * tanh(c_t)$$

输出门o_t：决定模型要将单元状态中的什么信息传递给隐藏状态

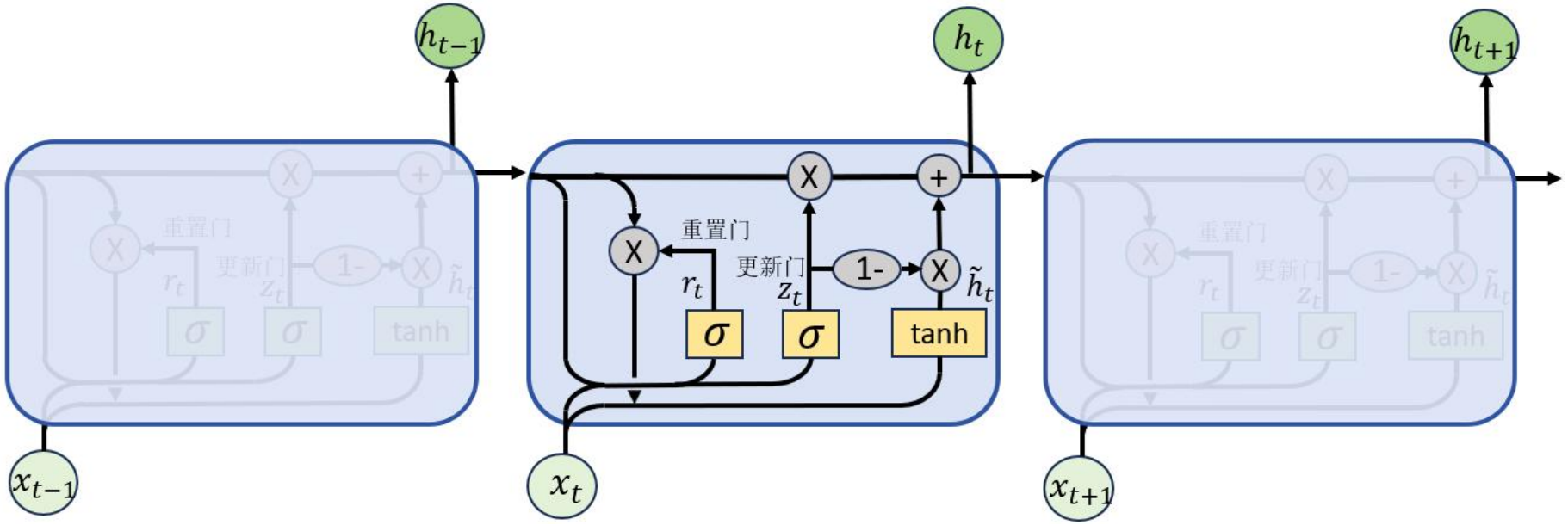
例如，当输出门o_t的值接近1时，就能够将单元内的信息传递给隐藏状态，而当输出门o_t的值接近 0 时，则不需要更新隐藏状态

目录

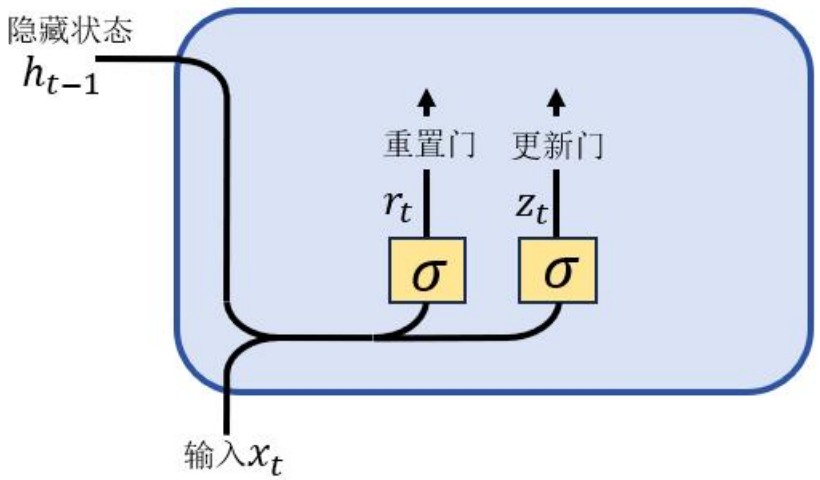
- 自然语言处理概述
- 词嵌入
  - 独热向量
  - word2vec
    - 跳元模型
    - 连续词袋模型
- 循环神经网络（RNN）
- 长短期记忆网络（LSTM）
- 门控循环单元（GRU）

门控循环单元（GRU）

- 门控循环单元的提出同样是为了解决反向传播中的梯度问题以及长期记忆问题，但相比于LSTM，GRU能在提供同等效果的同时有更快的计算速度
- GRU结构更简单，主要包括**重置门、更新门**两个门结构，**候选隐藏状态**以及**隐藏状态更新**两个主要步骤



门控循环单元（GRU）



1. 重置门和更新门

重置门：$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_x^{(r)}\boldsymbol{x}_t + \boldsymbol{W}_h^{(r)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(r)})$

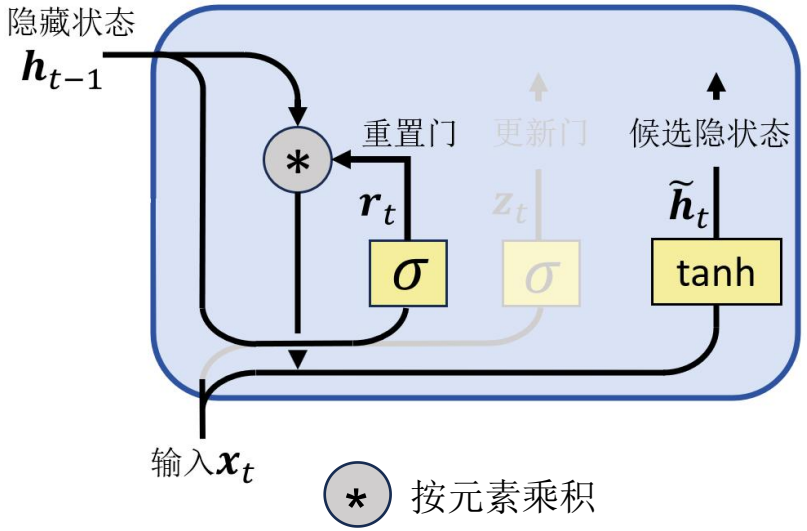
更新门：$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_x^{(z)}\boldsymbol{x}_t + \boldsymbol{W}_h^{(z)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(z)})$

两个门同样起到**选择性地让信息通过**的作用

重置门：决定隐藏状态中的什么信息需要保存。

更新门：决定新的隐藏状态多少来自候选隐藏状态多少来自旧隐藏状态。

门控循环单元（GRU）

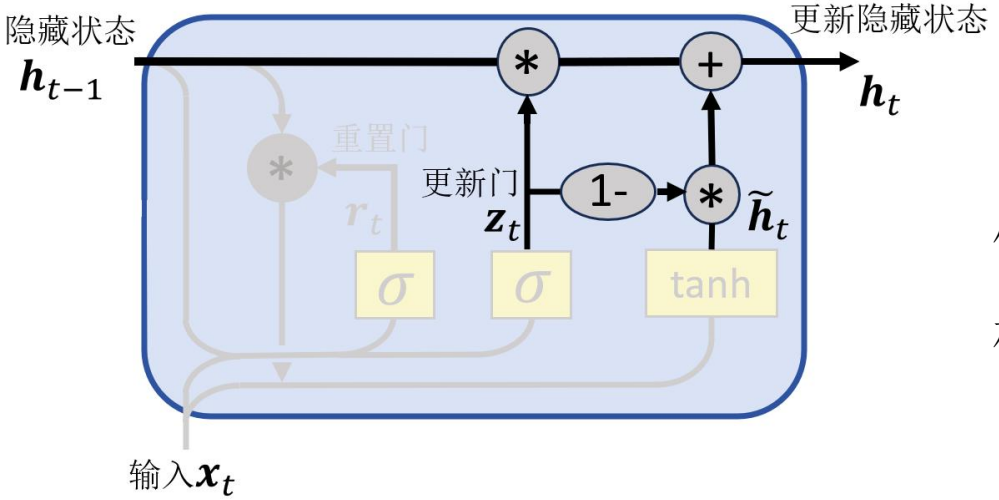


2. 候选隐藏状态

$$\tilde{h}_t = Tanh(W_x^{(h)} x_t + W_h^{(h)}(r_t * h_{t-1}) + b^{(h)})$$

重置门r_t：决定隐藏状态中的什么信息需要保存。

例如，r_t接近1时，模型就接近一个普通的循环神经网络。当r_t接近0时，则上一时刻的隐藏状态接近被忽略，候选隐藏状态是将x_t输入到线性层的结果。



3. 隐藏状态更新

$$h_t = z_t * h_{t-1} + (1 - z_t) * \tilde{h}_t$$

更新门z_t：决定新的隐藏状态多少来自候选隐藏状态多少来自旧隐藏状态

例如，当z_t接近1时，h_t就接近于h_{t-1}，模型就倾向于保留旧状态。

相反，当z_t接近0时，h_t就接近于$\tilde{h}_t$

谢谢！
Thanks!