# 机器学习

## 第九讲: 语言模型

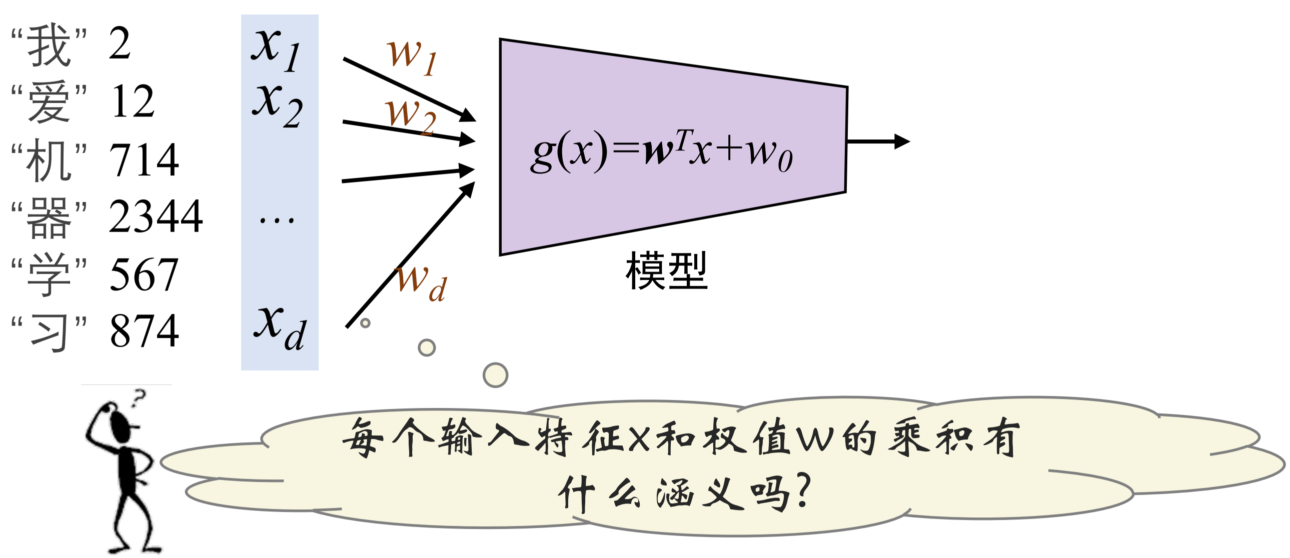
顾小东
上海交通大学计算机学院

# 本讲提纲

- **词向量**
  - 动机
  - 原理
  - 典型模型
- **语言模型**
  - 核心概念
  - 计算原理
- **循环神经网络**
  - 设计动机
  - 原理与应用

# 关于单词的两个问题

- 单词在计算机中如何表示?
  - 数字编码 (ASCII, id, etc.)
  - 举例: “hotel”:11, “conference”:81, “motel”: 1203, …
- 单词在机器学习模型中如何表示?



# 尝试用向量表示单词

- 单词可以表示成一个个**独热(one-hot)**向量:

一个位置是1, 其余维都是0

<u>独热编码 (One-Hot Encoding)</u>

hotel = [ 1  0  0  0  0  0  0  0  0]

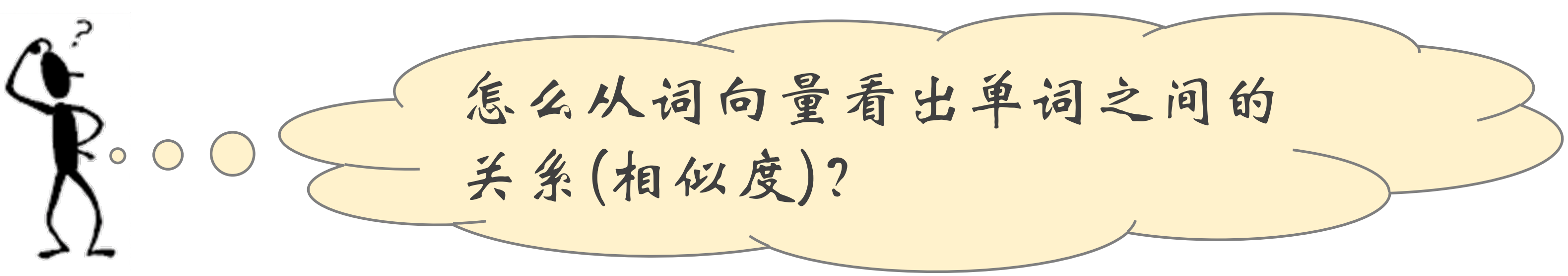
flower = [ 0  1  0  0  0  0  0  0  0]

tree = [ 0  0  0  1  0  0  0  0  0]

motel = [ 0  0  0  0  0  0  0  1  0]

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

向量维度 = 词汇表单词数 (e.g., **500,000**)



# 词嵌入技术(Word Embedding)

- 将单词表示为低维、浓缩的向量，向量之间的关系刻画了单词的语义相似度。

独热编码

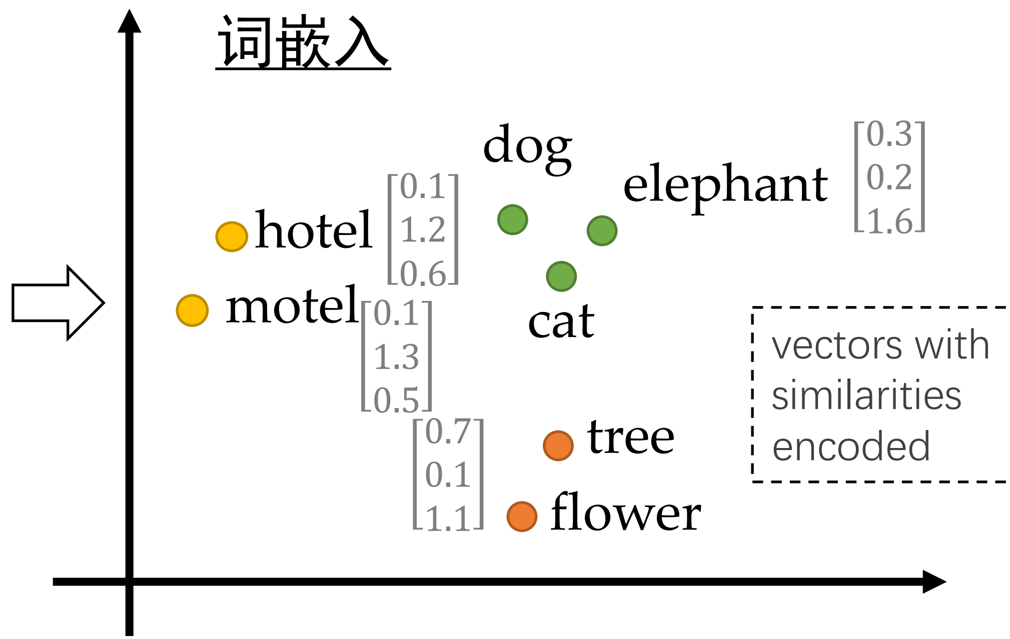
hotel = [ 1 0 0 0 0]

flower = [ 0 1 0 0 0]

business = [ 0 0 1 0 0]

motel = [ 0 0 0 1 0]

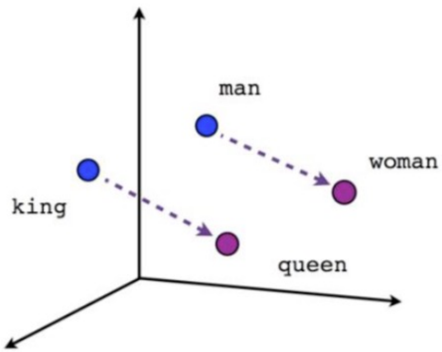
elephant = [ 0 0 0 0 1]



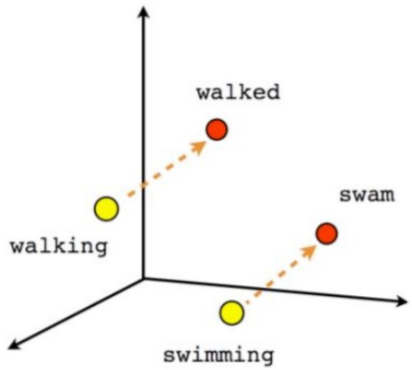

**Note**: word vectors are sometimes called word embeddings or word representations. They are distributed representations.

# 词嵌入技术(Word Embedding)

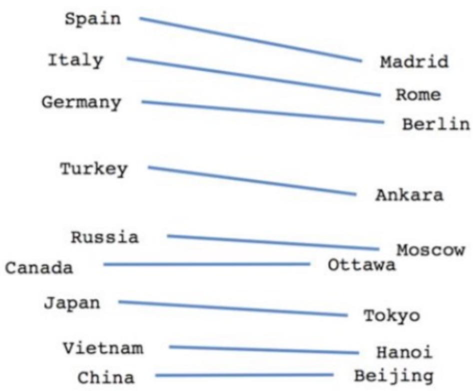
- **举例：**使用词嵌入技术可以刻画单词之间的丰富语义关系。




Male-Female




Verb tense




Country-Capital

# Word2Vec技术

Mikolov et al. 2013

- 根据单词在文本中的位置自动学习词向量的技术.

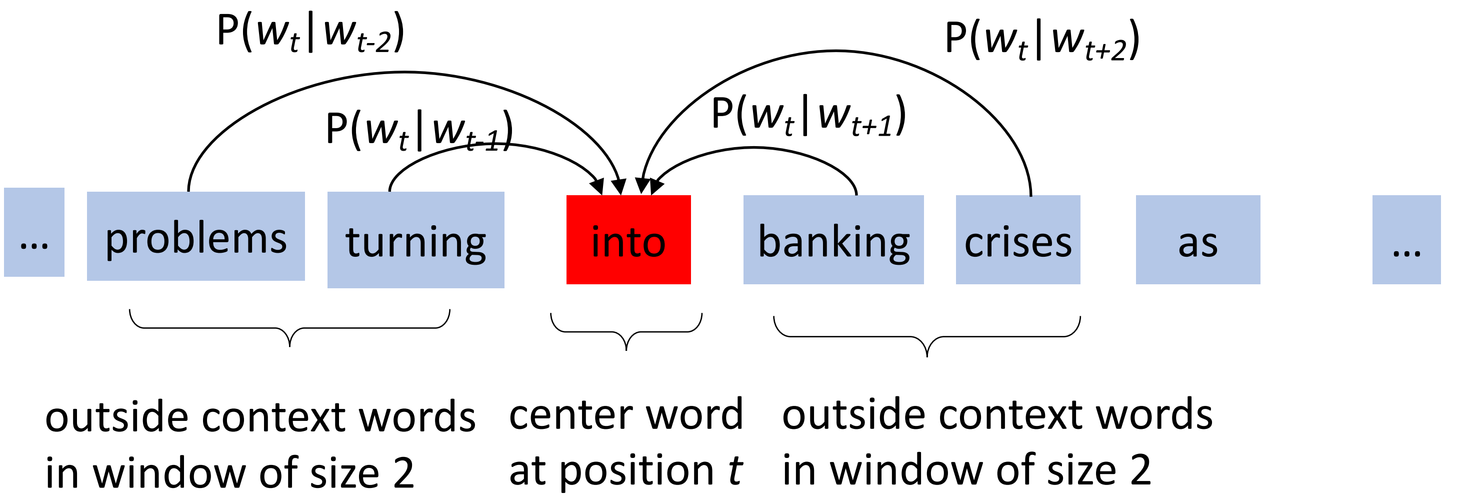
**Idea:**

- 收集一个巨大的文本库。
- 对词汇表中的每个单词赋予一个可学习的向量；
- 遍历文本，每个位置t关注一个中心词w和上下文单词c ；
- 根据c和w的向量相似度计算它们出现的条件概率p(w|c) ；
- 调整c和w的词向量，使条件概率p(w|c)最大.

Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. NIPS 2013.

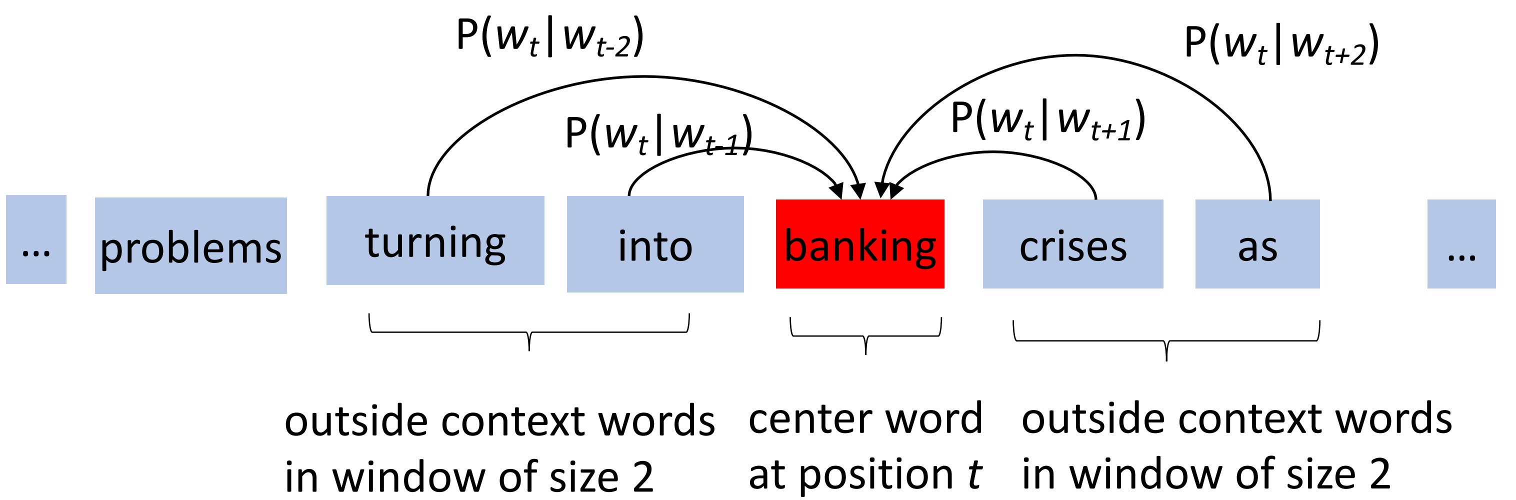
# Word2Vec技术

例: $P(w_t|w_{t+j})$



# Word2Vec技术

例: $P(w_t|w_{t+j})$



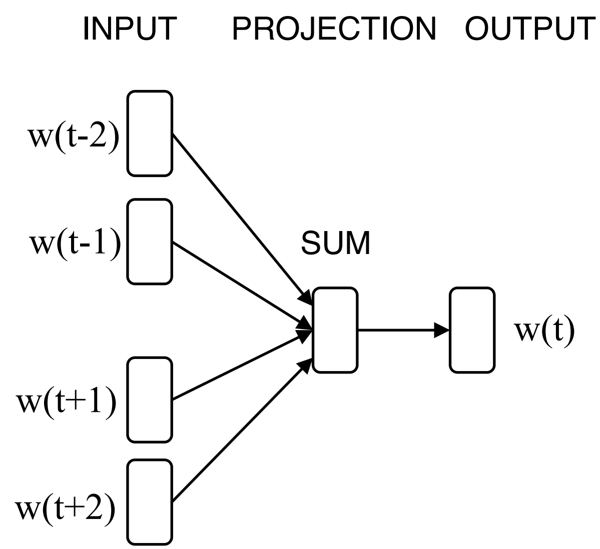
# CBOW模型 Mikolov's 2013

- **CBOW**: 根据上下文单词的词向量预测中心词。**完型填空**

$$P(w_t \mid w_{t-k}, \ldots w_{t-1}\ w_{t+1} \ldots w_{t+k}) = \text{softmax}\left(\mathbf{NN}\left(V(w_{t-k}) + \ldots + V(w_{t+k})\right)\right)$$

**使用神经网络实现CBOW:**

- **输入层**：将每个单词的独热向量转成词向量；
- **映射层**：将输入词向量进行聚合.
- **输出层**：根据聚合后的上下文向量预测中心词w(t)。

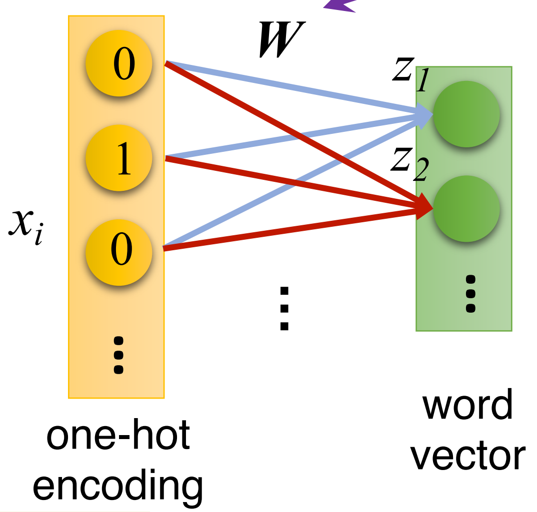



CBOW = Continuous-Bag-of-Words

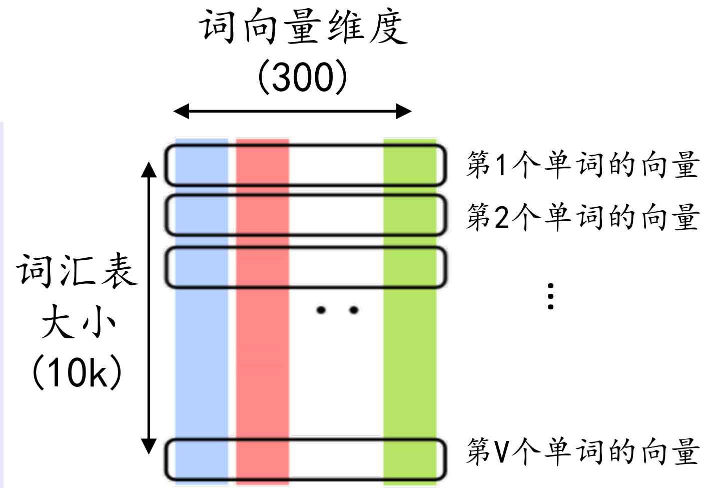
# 输入层：将任意单词转成词向量

- **实现方式:** 将原始词表示(独热向量)进行线性变换，得到词向量：

$$z_i = \boldsymbol{W} x_i = \boldsymbol{W}_i$$



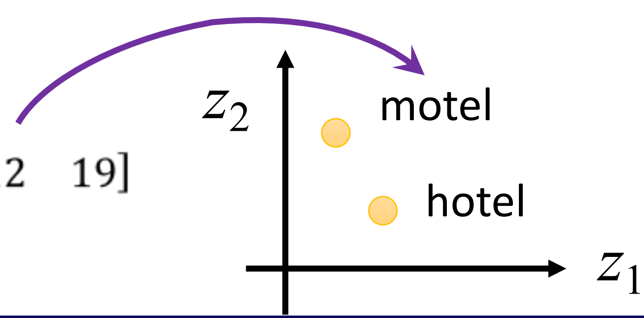

这里的权值矩阵$\boldsymbol{W} \in R^{|V| \times d}$比较特殊，实际上是一个查找表(**lookup table**)，每一行对应词汇表中对应位置单词的词向量。




例子

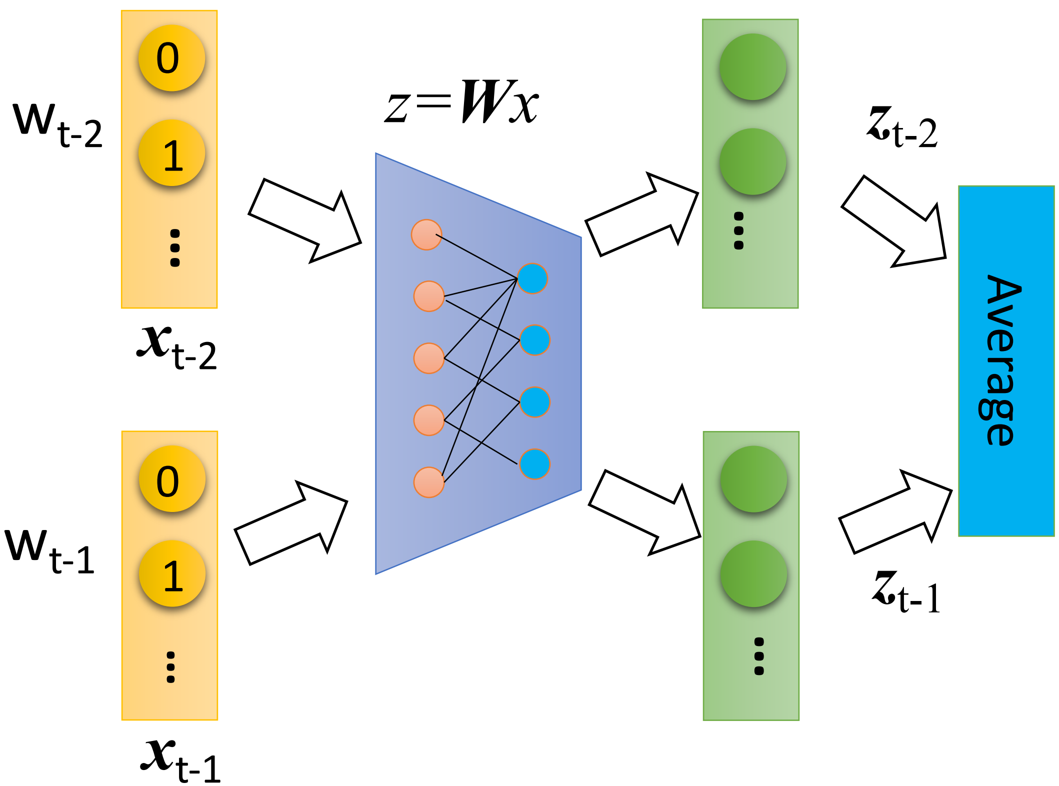
$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

“motel”



# 投影层：聚合上下文向量

- 通过取平均或求和，将所有上下文向量聚合成一个向量。

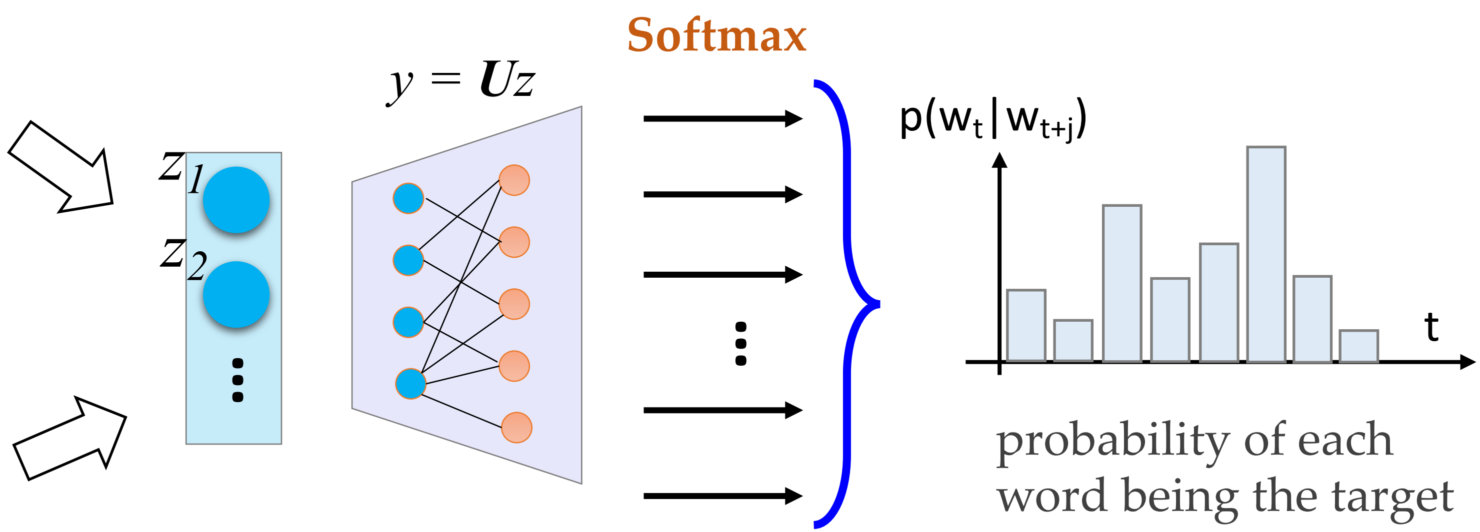



$$z = \frac{1}{2|c|} \sum_{-|c| \le j \le |c|, j \ne 0} z_{t+j}$$

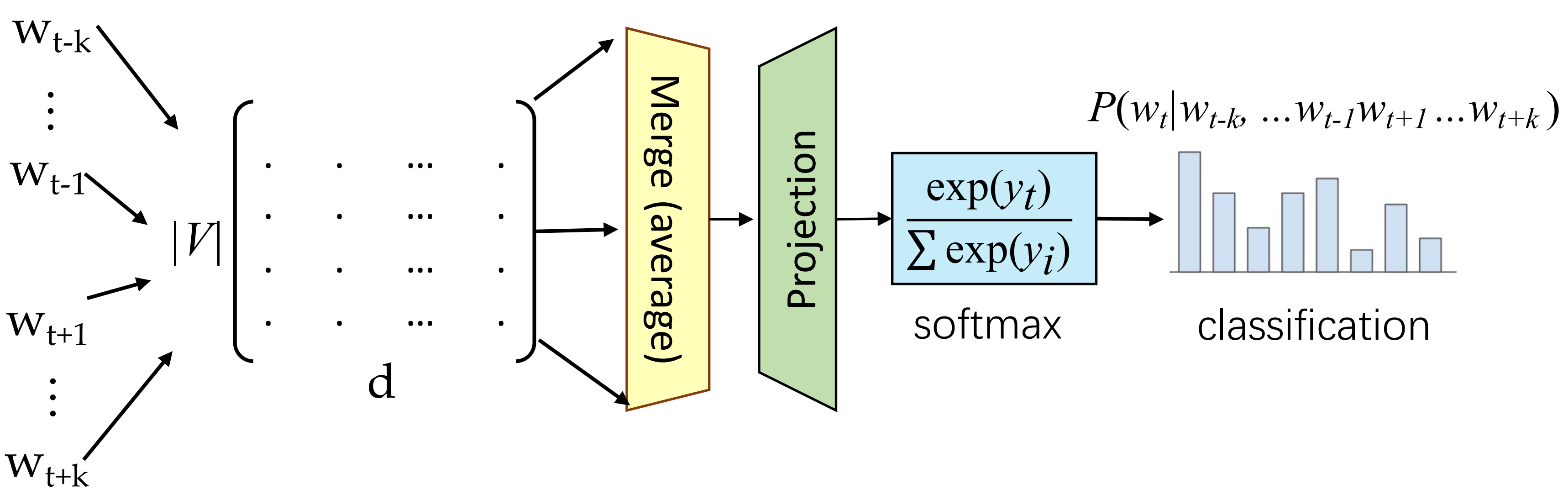
# 输出层：预测中心词

- 根据聚合后的向量，预测中心词。
- 实际上是一个关于词汇表中单词的分类问题。

$$\mathrm{P}\left(w_{\mathrm{t}} \middle| w_{\mathrm{t}-|\mathrm{c}|}, \ldots, w_{\mathrm{t}-1}, w_{\mathrm{t}+1}, \ldots, w_{\mathrm{t}+|\mathrm{c}|}\right) = \frac{\exp(y_{\mathrm{t}})}{\sum_{\mathrm{i}=1}^{|\mathrm{V}|} \exp(y_{\mathrm{i}})}$$



# 模型总体框架



$w_{t-k}$
$w_{t-1}$
$w_{t+1}$
$w_{t+k}$
$|V|$
d
Merge (average)
Projection
$\frac{\exp(y_t)}{\sum \exp(y_i)}$
softmax
$P(w_t|w_{t-k}, \ldots w_{t-1} w_{t+1} \ldots w_{t+k})$
classification

# 训练方法

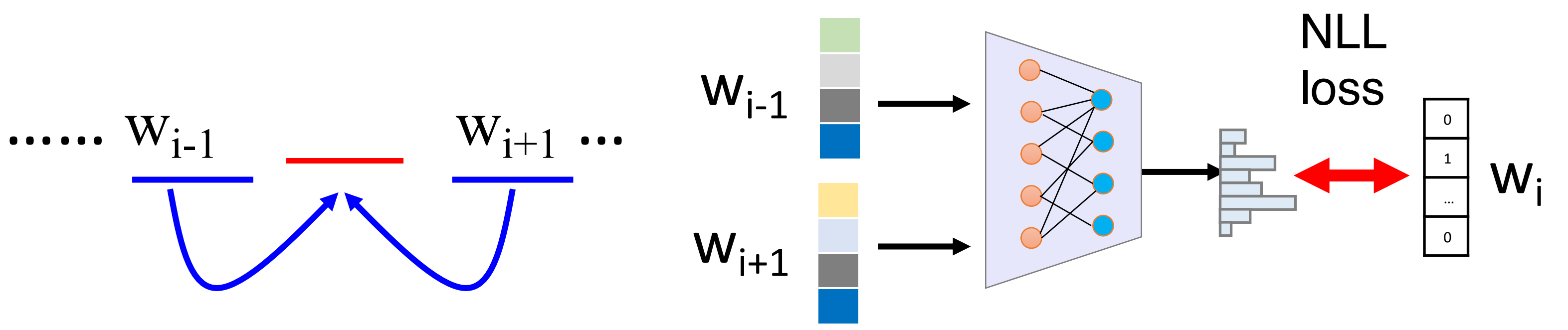
- 给定数据集 $D = \{w_1, w_2, \ldots, w_N\}$, 最小化负对数似然(**negative log likelihood,** NLL) 损失函数:

$$L(\boldsymbol{W}, \boldsymbol{U} \mid D) = -\frac{1}{N}\sum_{t=1}^{N} \log p(w_t | w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k})$$

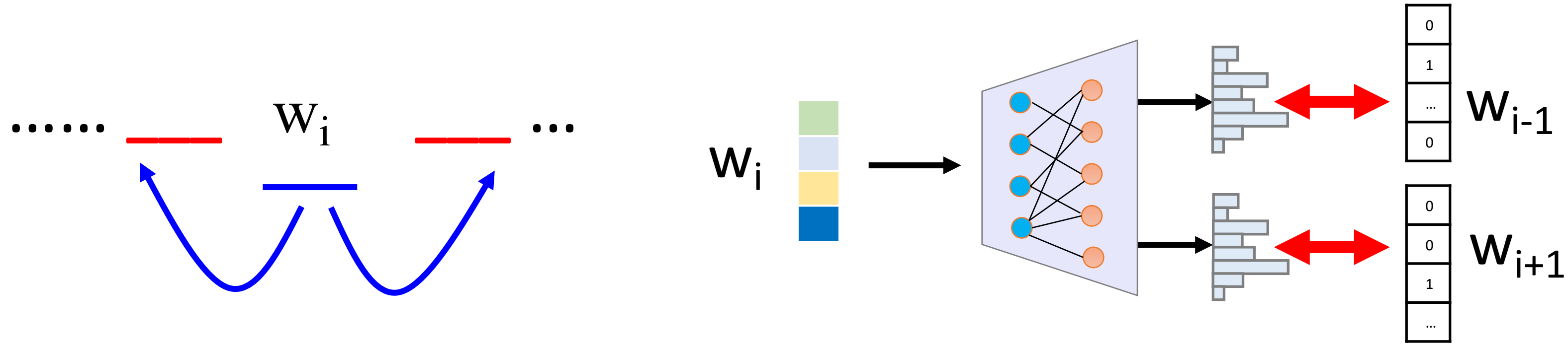
- 使用梯度下降最小化损失函数。

# 其他词向量模型

- **CBOW** 根据上下文预测中心词




- **Skip-gram** 根据中心词预测上下文



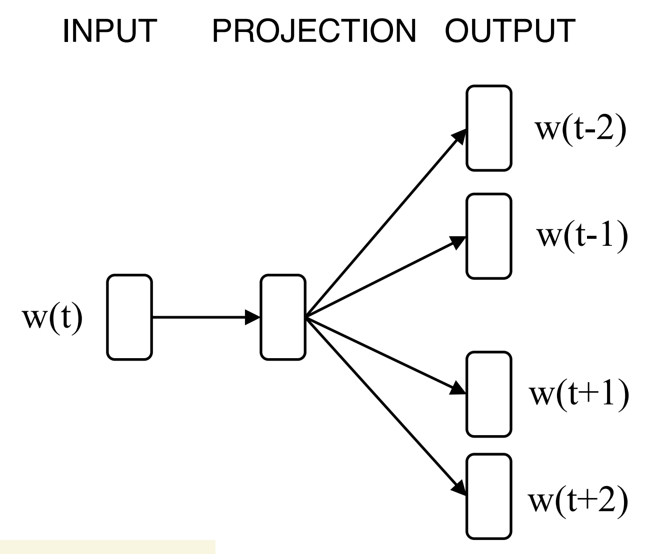
# Skip-Gram模型

- We seek a model for $P(w_{t+j}|w_t)$.

$$P(w_{t+j}|w_t) = \frac{\exp(y_{t+j})}{\sum_{i=1}^{|V|}\exp(y_i)}$$

$$y = \boldsymbol{U}z$$

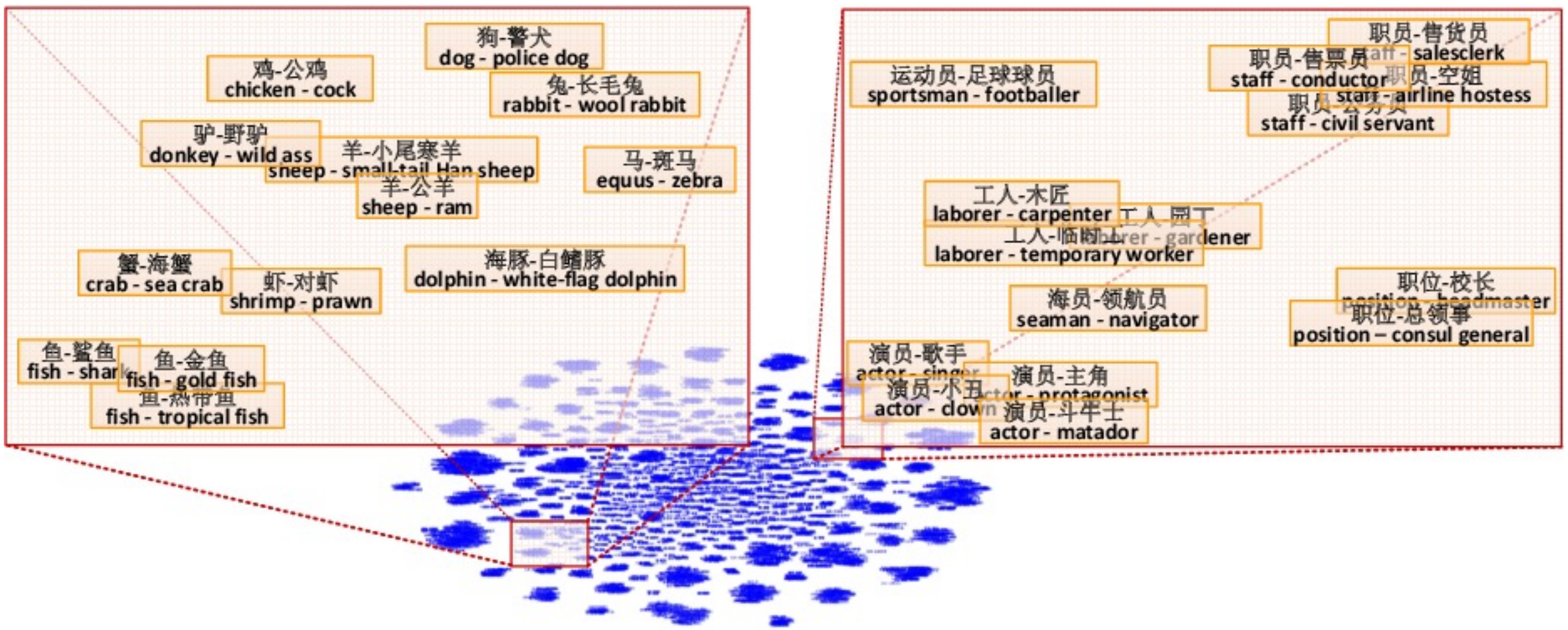
$$z = \boldsymbol{W}x$$




$$L(W,U|\chi) = -\frac{1}{N}\sum_{t=1}^{N}\sum_{-c\le j\le c, j\ne 0}\log p(w_{t+j}|w_t)$$

# 实例展示

- Word2vec maximizes an objective function by putting similar words nearby in space.




Fu, Ruiji, et al. "Learning semantic hierarchies via word embeddings.”*ACL* 2014.

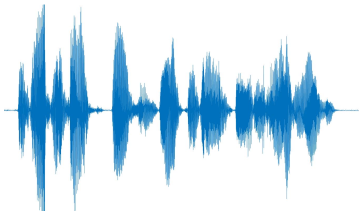
# 编程实践

**Tutorial: 使用python学习词向量**

# 从“背单词”到“学语言”

无处不在的语言数据

- Audio:



- Word:

自然，语言

- Sentence:

我爱自然语言处理

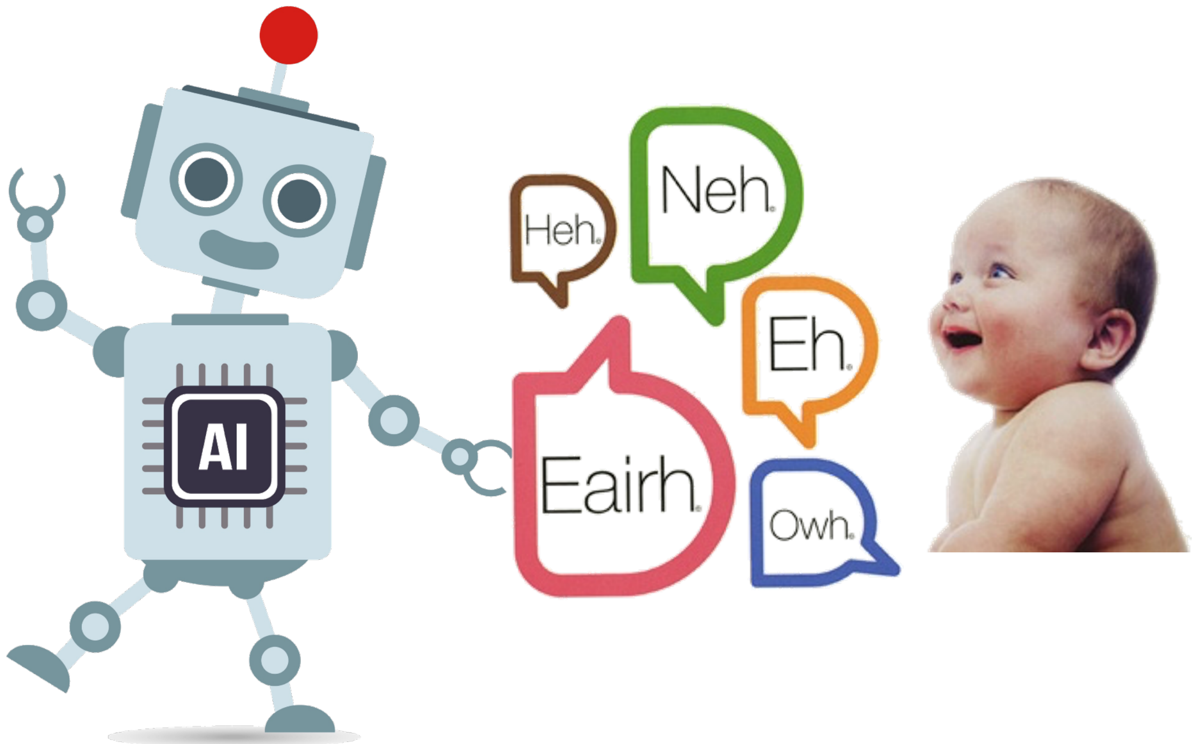
- Program:

```
for (int i = 0; i < 10 ; i ++ ) {
```

...

# 如何让计算机学会人类语言？

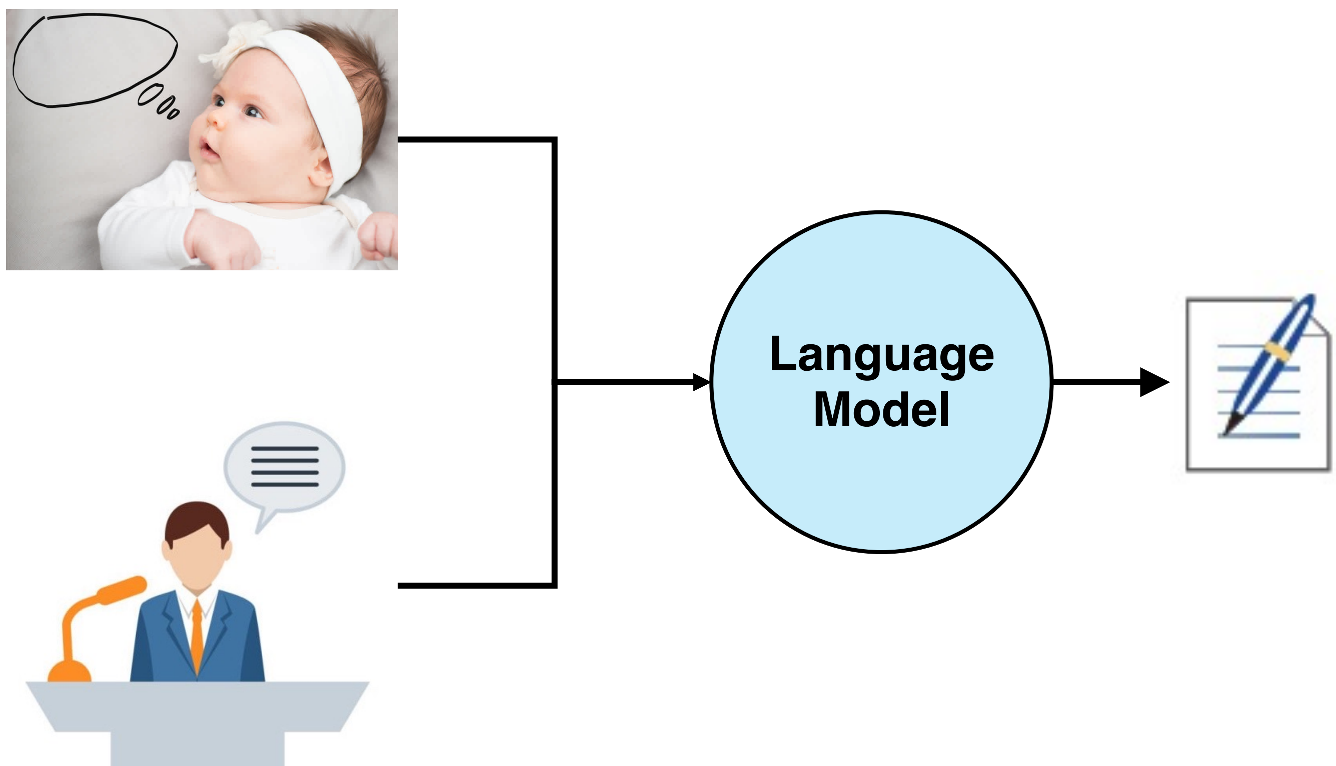
- 和教婴儿一样从头教计算机讲话？




But wait, how to measure the linguistic ability of computers?

# 如何衡量语言能力？

- How proficiency do you acquire a language?



# 语言模型

- 语言模型是一种概率模型，它刻画了给定的字符串有多大可能在某个语言中出现。
- 对任意字符序列$x = (w_1, w_2, \ldots, w_N)$, **语言模型**定义为该序列在某个语言中出现的概率:

$$p(x) = p(w_1, w_2, \ldots, w_N)$$

# 语言模型的计算

- 如何计算概率$P(w_1, ..., w_N)$?

$p$(我爱机器学习) = ?

- 根据概率的链式法则:

$$p(w_1,\dots,w_N) = p(w_1)p(w_2|w_1)\dots p(w_N|w_1,\dots,w_{N-1})$$

p(我爱机器学习) = p(我)p(爱|我)p(机|我爱)p(器|我爱机)p(学|我爱机器)p(习|~~我爱~~机器学)

- 马尔科夫假设: (只考虑前n−1个单词)

$$p(w_i|w_1,\dots,w_{i-1}) = p(w_i|w_{i-n+1},\dots,w_{i-1})$$

$p$(习|我爱机器学) ≈ $p$(习|机器学) ≈ $p$(习|学)

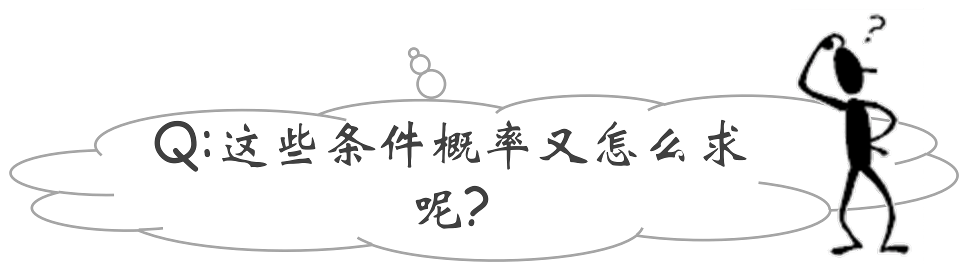
# 语言模型 = 预测下一单词

- **语言模型归根结底是求解以下问题:**
  - 给一个序列的t-1个单词 $w_1, \ldots, w_{t\text{-}1}$
  - 预测下一个单词出现的概率 $p(w_t|w_1, \ldots, w_{t\text{-}1})$

“我 爱 机 器 学 习”

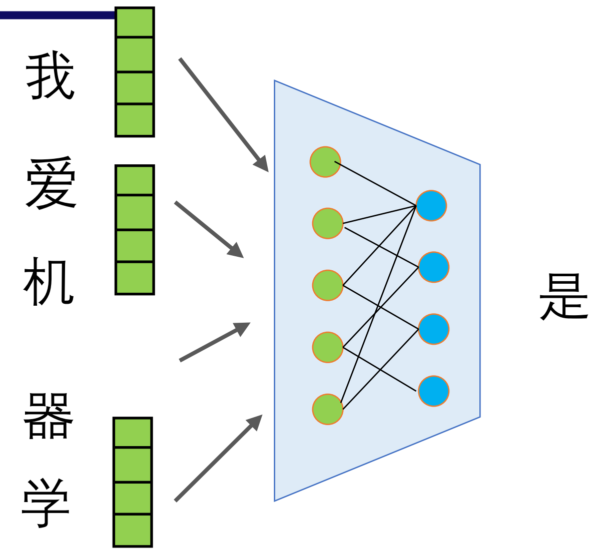
given these words

predict the next word



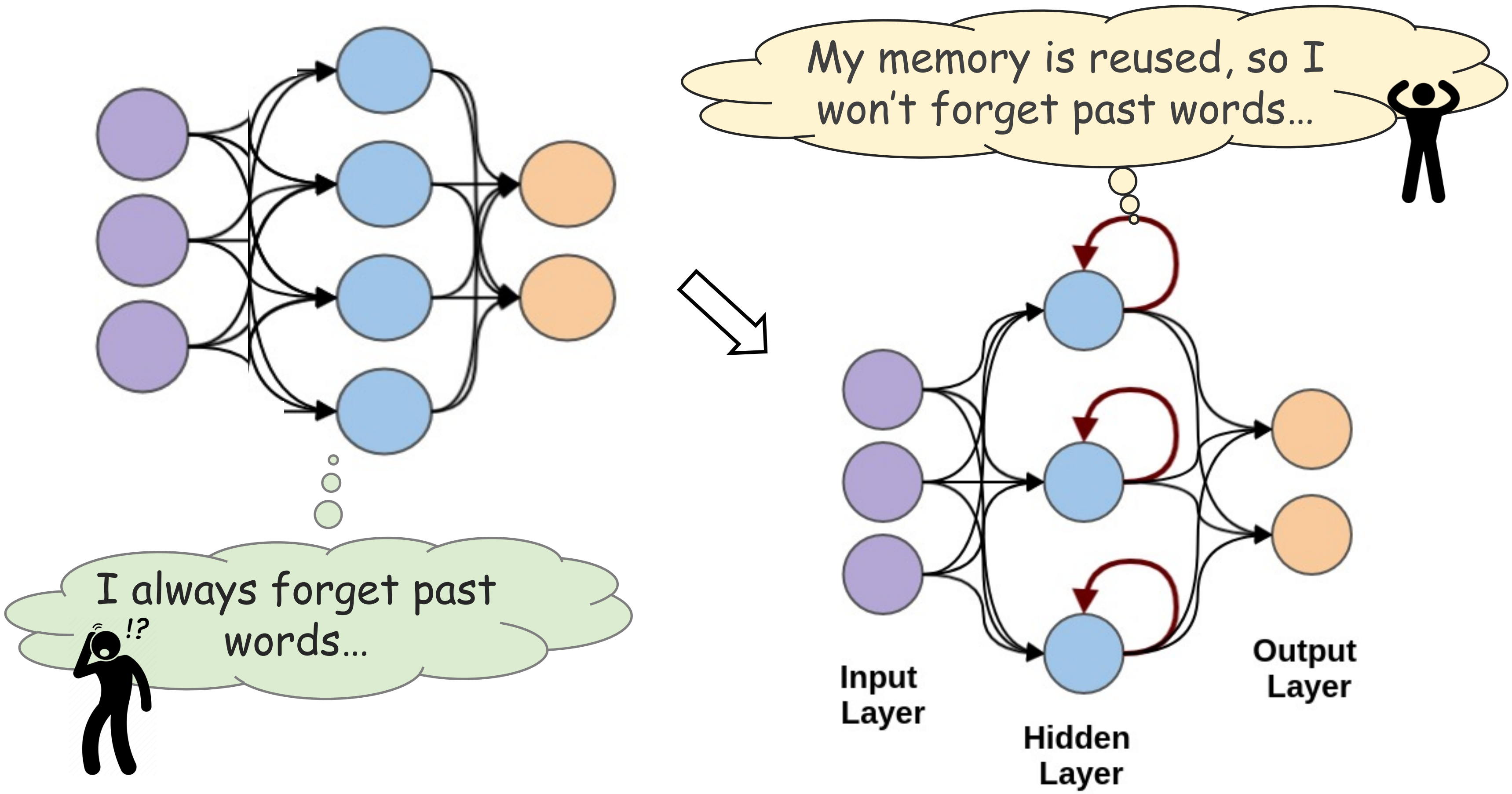
# 神经网络语言模型

- 设计神经网络对语言(序列)进行建模
- **Recurrent Neural Networks** (RNNs) as an ideal approach for sequence modeling.



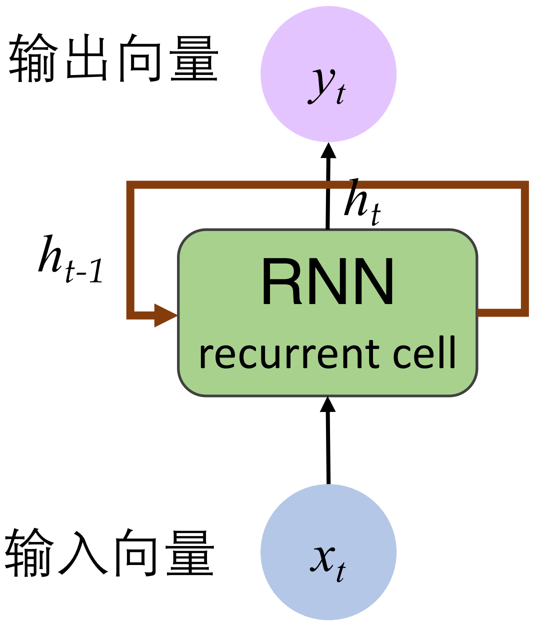
# 循环神经网络

**<u>核心思想</u>:** 复用隐含层来记住过去输入的单词。



# 循环神经网络

**基本结构**: 将上一时刻的隐含状态$h_{t-1}$作为下一时刻的输入，实现记忆循环。




Apply a recurrent relation at every time step to process a sequence:

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

cell state

function parameterized by W

old state

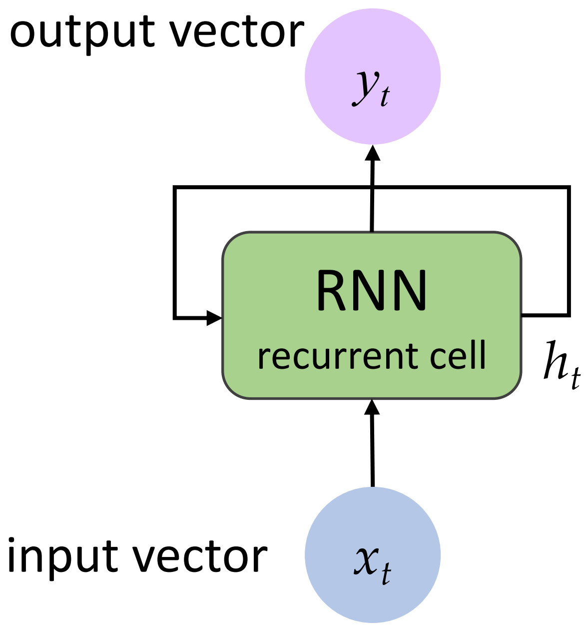
Input vector at time step t

Note: the same function and set of parameters are used at every time step

RNNs have a **state**, $h_t$, that is updated **at each time step** as a sequence is processed.

# 循环神经网络

## 状态更新过程




Output Vector

$$y_t = \boldsymbol{W}_{hy}^T h_t$$

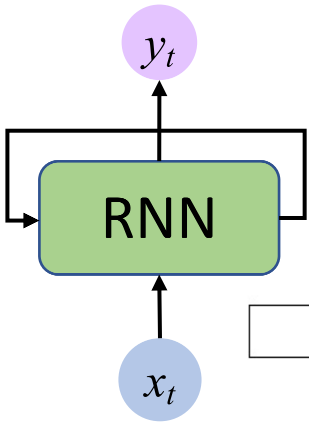
Update Hidden State

$$h_t = \tanh(\boldsymbol{W}_{\mathrm{hh}}^{\mathrm{T}} h_{t-1} + \boldsymbol{W}_{\mathrm{xh}}^{\mathrm{T}} x_t)$$

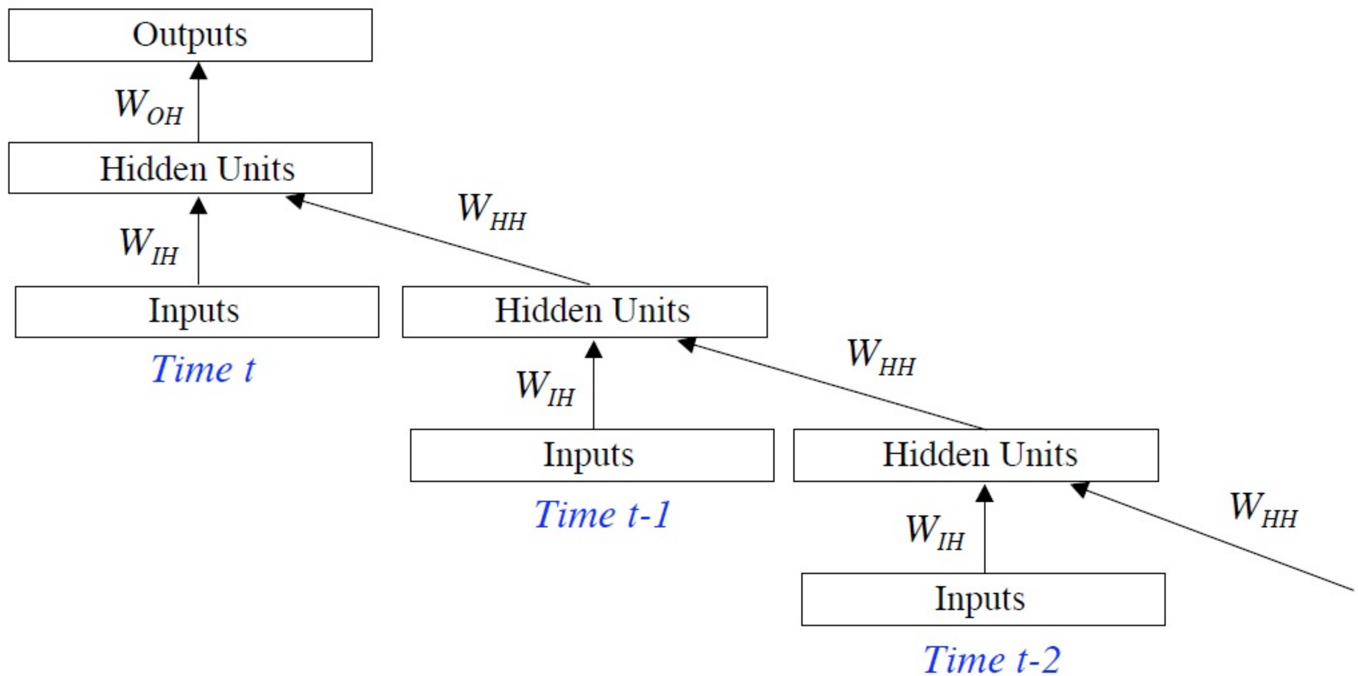
Input Vector

$$x_t$$

# RNNs: Computational Graph Across Time

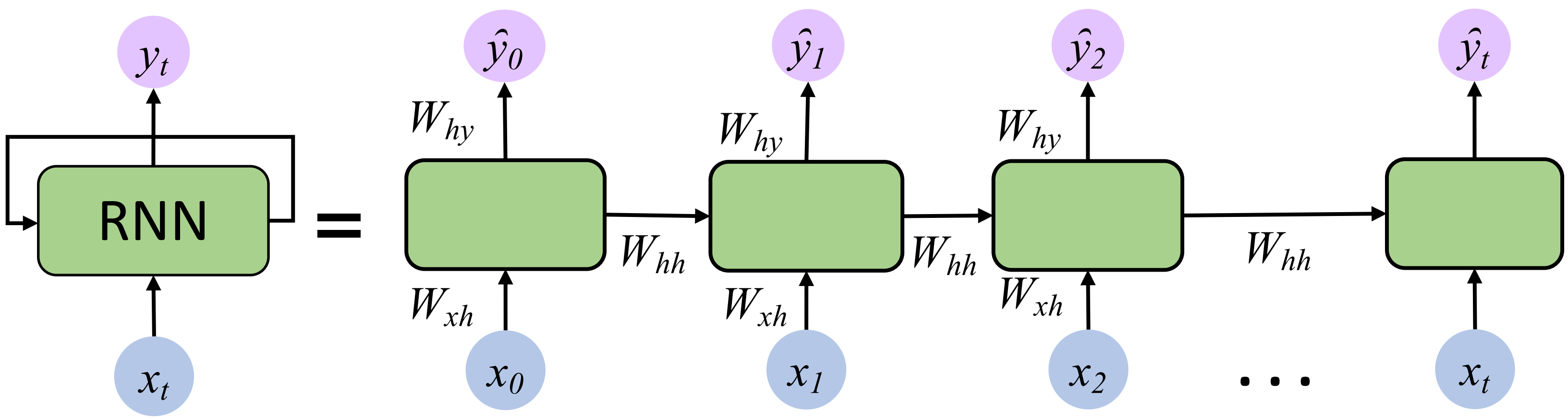



= Represent as computational graph unrolled across time



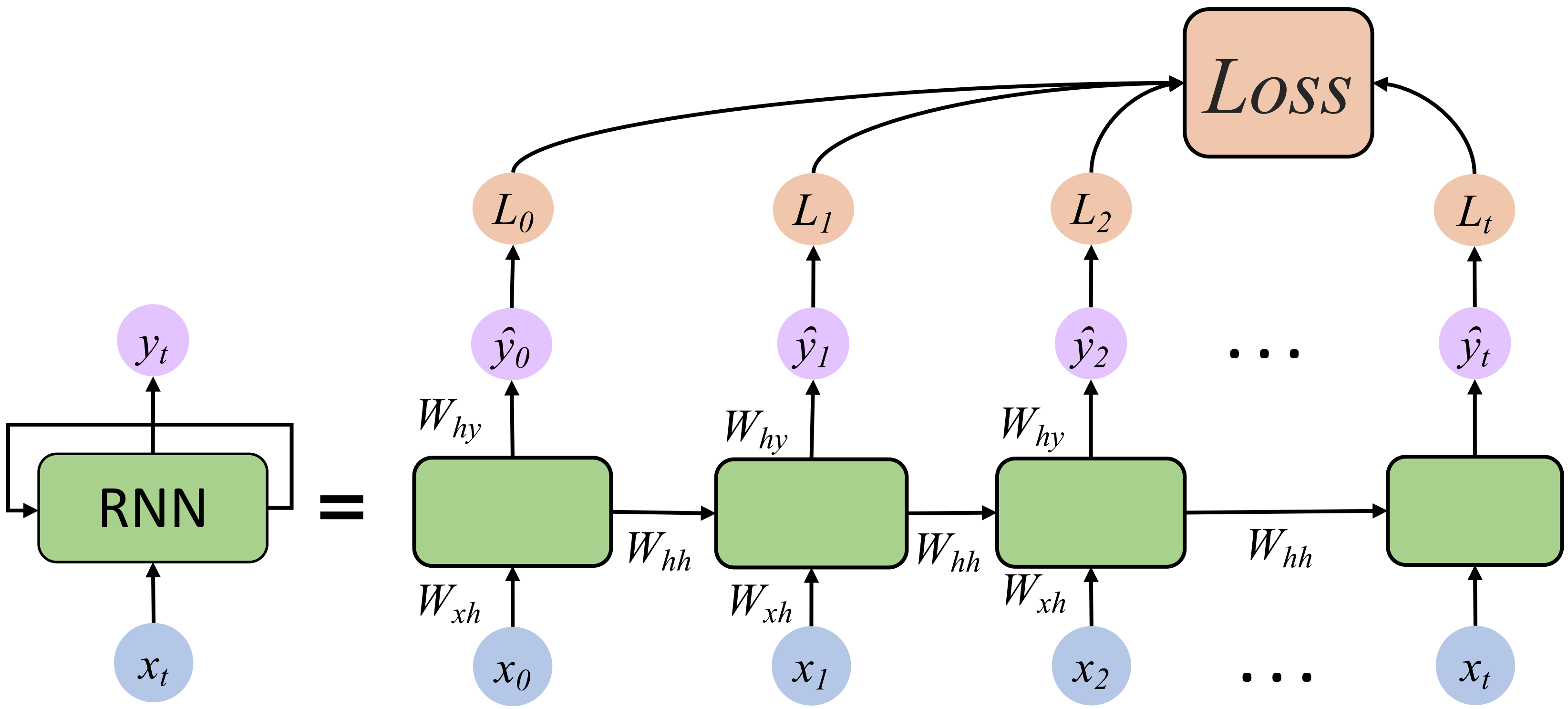
# RNNs: Computational Graph Across Time

- Re-use the same weight matrices at every time step



# RNNs: Computational Graph Across Time

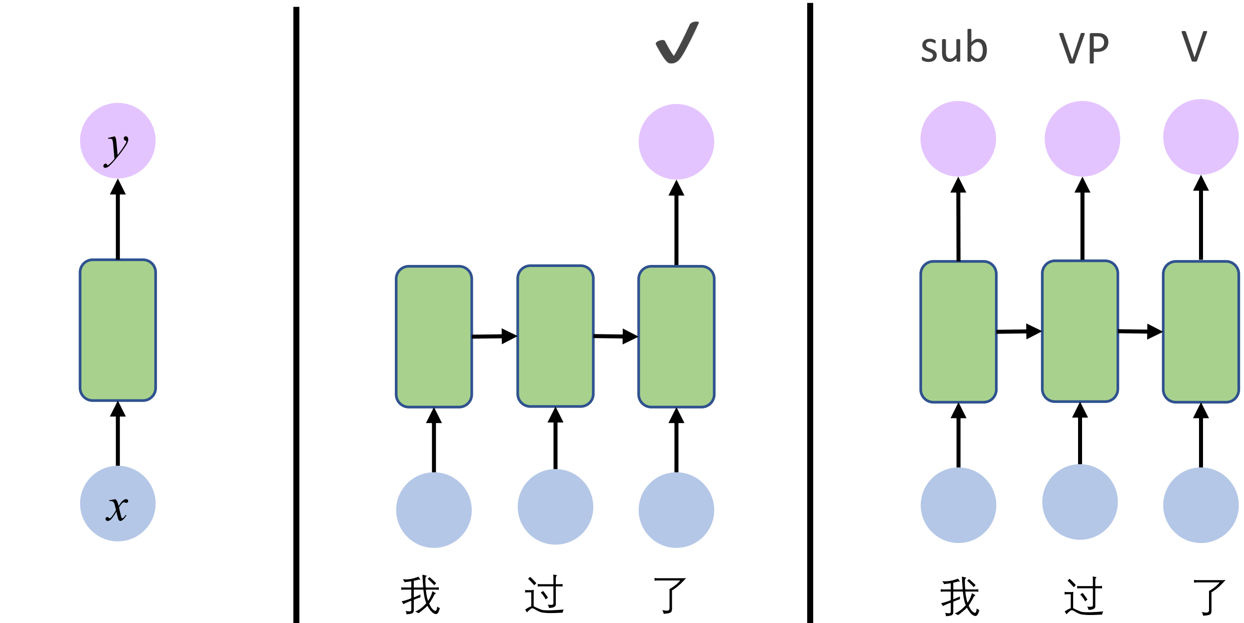
→ forward pass



# RNN的应用

# 应用模式

- 根据输入输出形式的不同，有两种使用模式



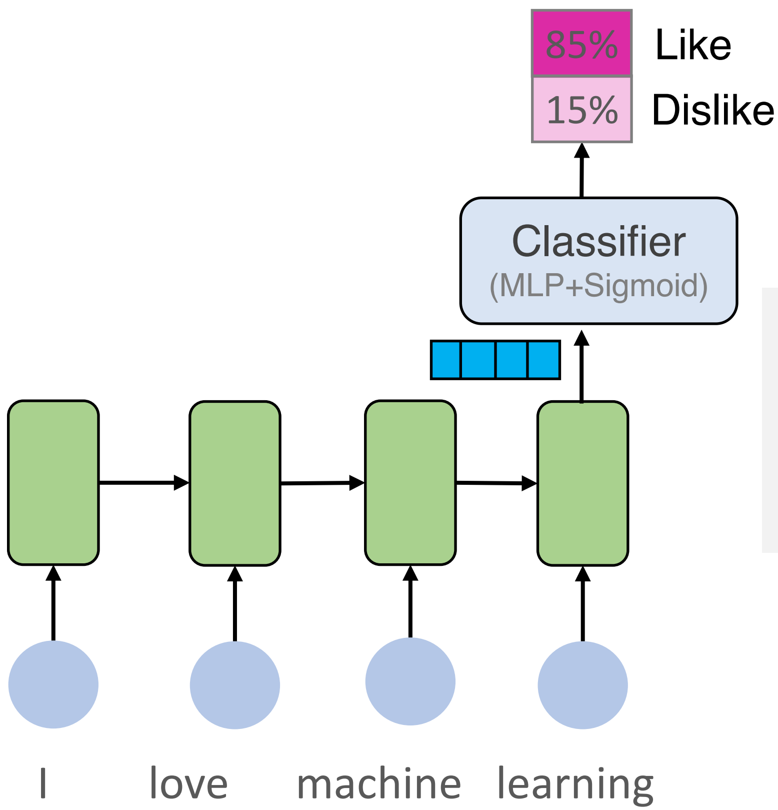

**One to One**
"Vanilla" neural network

**Many to One**
e.g., sentiment classification

**Many to Many**
e.g., information extraction

...and many other architectures and applications

# 应用一：序列分类

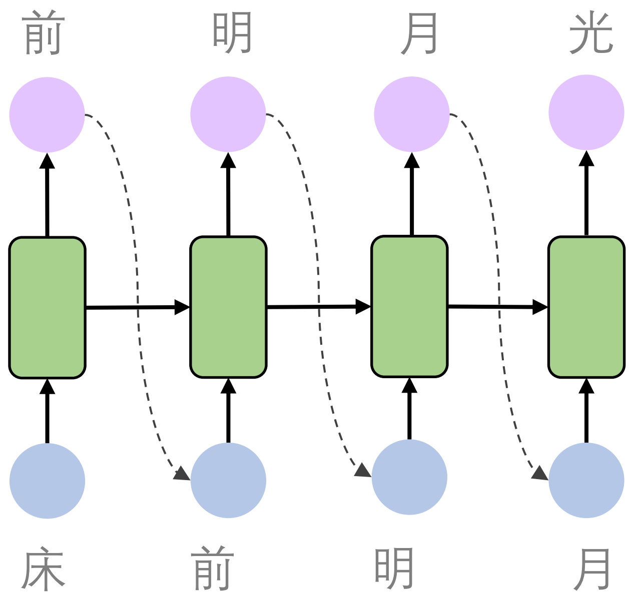



**Input:** a sequence of words.

**Output:** probability of each class that the sequence belongs to

# 应用二：序列生成

- 举例：AI作诗

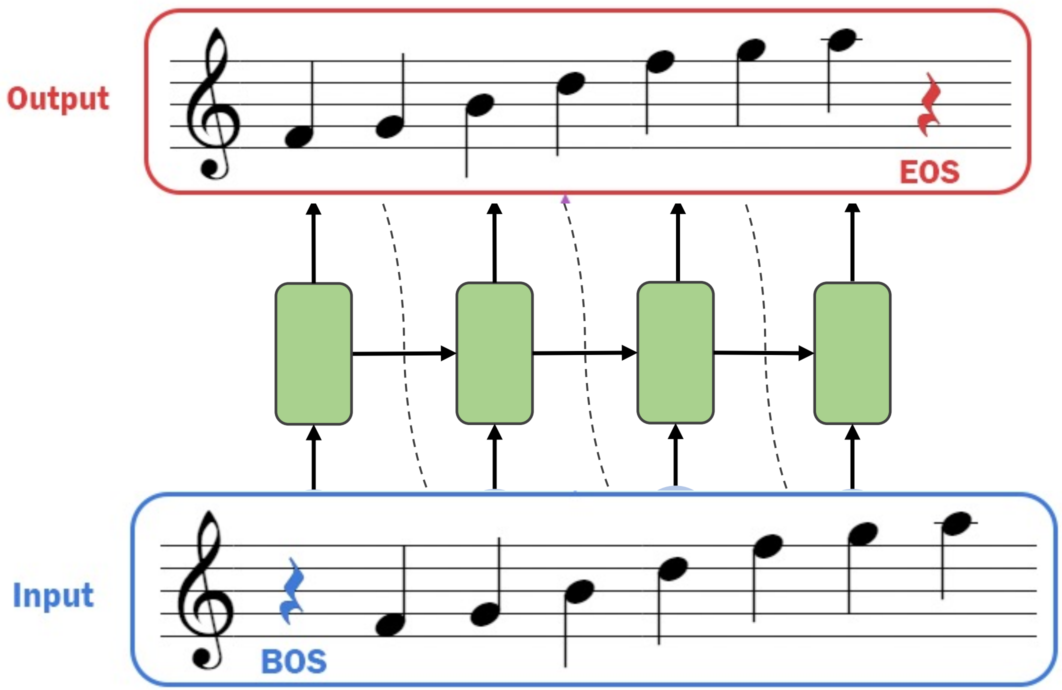



**Input:** n-1 poetry characters

**Output:** next character in a poetry

# 应用二：序列生成

- **<u>举例</u>**：音乐生成




**Input:** sheet music

**Output:** the next character in sheet music

# TIME for Coding

## Tutorial: poetry generation using Pytorch

https://gitee.com/wannabe-9/LSTM_poem1

https://zhuanlan.zhihu.com/p/138270447

https://www.kaggle.com/code/jiaminggogogo/ancient-chinese-poem-generation-rnn/notebook