

# 词向量

# 如何表示字词的含义?

## ● 早期探索

- WordNet: 一个大型英语词汇数据库，将单词之间的关系用概念网络表示，单词之间的主要关系是同义词 (Synonym) 和上位词 (Hypernym)。

PRINCETON UNIVERSITY

## WordNet

A Lexical Database for English

What is WordNet

People

News

Use WordNet Online

Download

Citing WordNet

License and  
Commercial Use

Related Projects

Documentation

Publications

Frequently Asked  
Questions

### What is WordNet?

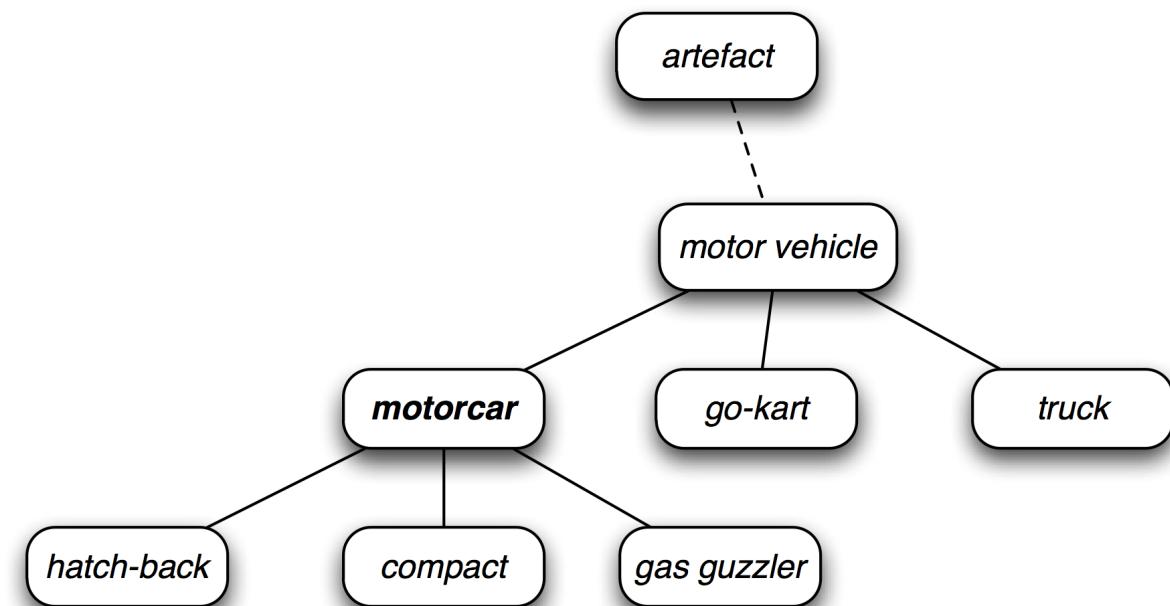
Any opinions, findings, and conclusions or recommendations expressed in this material are those of the creators of WordNet and do not necessarily reflect the views of any funding agency or Princeton University.

When writing a paper or producing a software application, tool, or interface based on WordNet, it is necessary to properly cite the source. Citation figures are critical to WordNet funding.

### About WordNet

WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions. First, WordNet interlinks not just word forms—strings of letters—but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated. Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity.



# 如何表示字词的含义?

## ● 早期探索

e.g., synonym sets containing “good”:

```
from nltk.corpus import wordnet as wn
poses = { 'n': 'noun', 'v': 'verb', 's': 'adj (s)', 'a': 'adj', 'r': 'adv' }
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
                          ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
...
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g., hypernyms of “panda”:

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(pandaclosure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

# 如何表示字词的含义？

## ● WordNet的主要问题

- 没有考虑语境和上下文（而语境、上下文在自然语言处理中非常重要）；
- 缺少最新词汇，难以做到实时更新（语言是动态的、不断发展的，每年都会出现很多新词汇）；
- 需要耗费人力进行创建、维护，工作量很大；
- WordNet的构建是主观的，而不是基于客观数据的；
- 较粗糙，难以精确地衡量词汇之间的相似度。

# 将字词表示为向量

## • 离散表示 (One-hot表示)

- 将每个词表示为One-hot向量，例如：

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

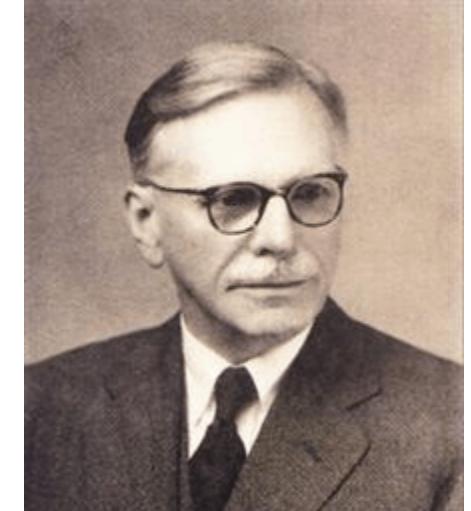
`hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0]`

- 向量的维度=词典中的词汇数量（50万以上）
- 主要问题：
  - 词汇数量很多，向量维度太高；
  - 难以计算词汇之间的相似性（每两个向量都是正交的）。

# 将字词表示为向量

## • 分布式表示（连续表示）

- 现代自然语言处理领域最成功的技术思想之一；
- 字词的含义与跟它经常一起出现的词密切相关 (J. R. Firth) ；
- 例如：
  - 当一个单词  $w$  在文本中出现时，其上下文 (Context) 就是与之一起出现的单词（在某个固定长度的窗口）；
  - 我们可以使用很多  $w$  的上下文构建  $w$  的向量表示；
  - 如在下面的上下文中都出现了 banking 这个词：



*You shall know a word by the company it keeps.*

——J. R. Firth, 1957

*...government debt problems turning into banking crises as happened in 2009...*

*...saying that Europe needs unified banking regulation to replace the hodgepodge...*

*...India has just given its banking system a shot in the arm...*

# 将字词表示为向量

## • 词向量

- 为每个单词构建一个密集向量 (Dense vector) —— 向量的每个元素都是一个浮点数；
- 在相似上下文中出现的单词，词向量也是“相似”的；
- 可以用向量点积计算词与词之间的相似度；
- 词向量 (Word vector) 也称为词嵌入 (Word embedding) 或者词的表示 (Word representation)。

$$\begin{aligned} \text{banking} &= \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix} & \text{monetary} &= \begin{pmatrix} 0.413 \\ 0.582 \\ -0.007 \\ 0.247 \\ 0.216 \\ -0.718 \\ 0.147 \\ 0.051 \end{pmatrix} \end{aligned}$$

# 将字词表示为向量

## • 词向量

*expect* =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

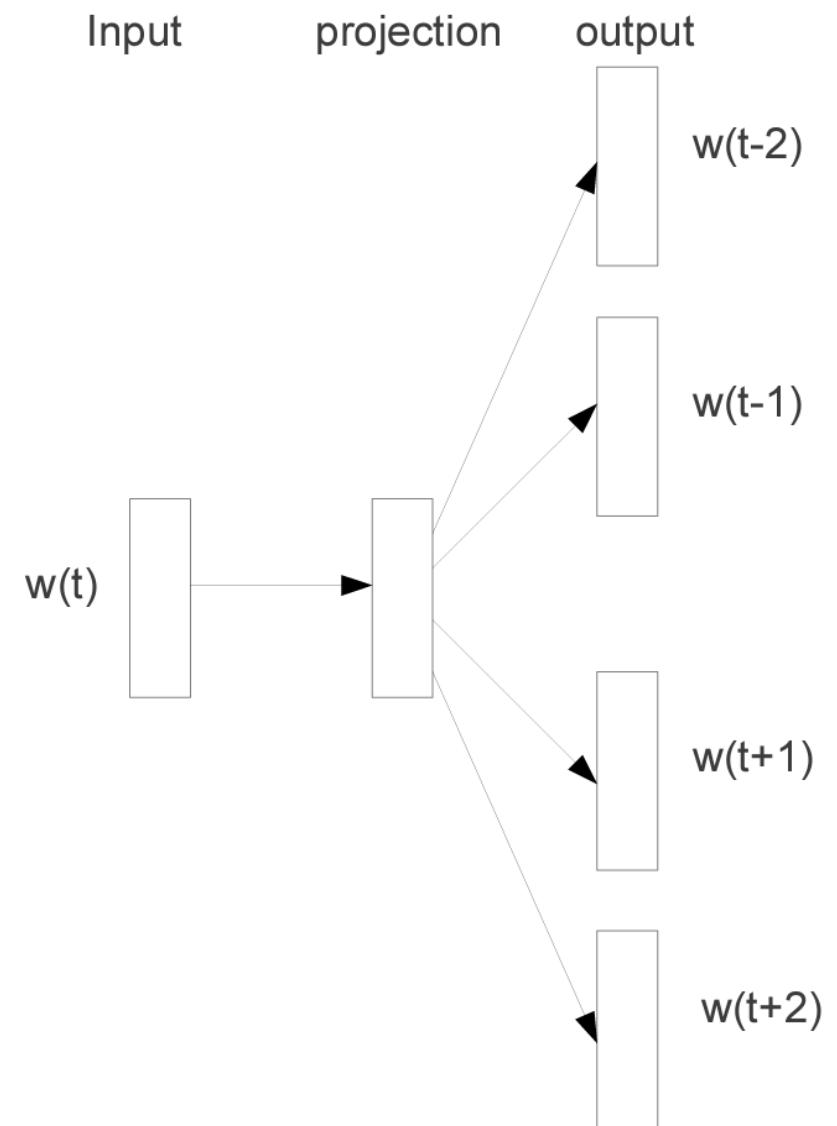
A 3D plot illustrating word embeddings. The vertical axis represents the third dimension of the vector space. The horizontal axes represent the first two dimensions. The words are positioned as follows:

- give, meet, say, think, expect, see, want, become, remain, be, are, is, were, was, being, been, had, has, have are clustered along the positive vertical axis.
- keep, make, get, continue, take are clustered along the negative vertical axis.
- come, go are at the top center.
- need, help are at the top right.

# Word2vec

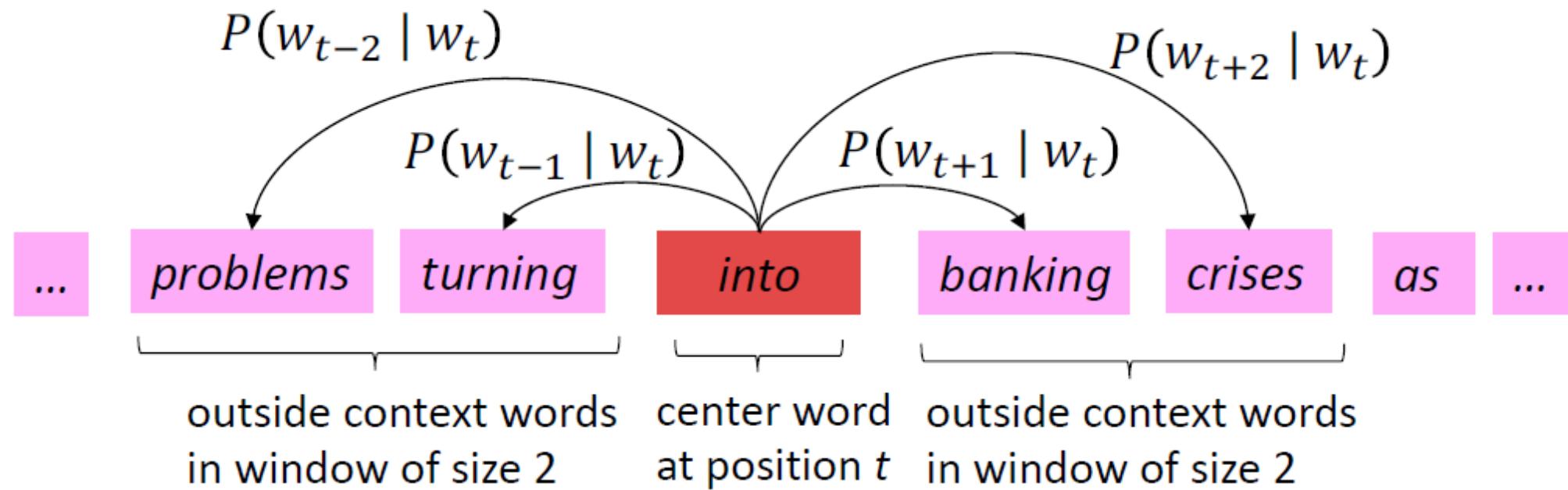
## • Word2vec

- Word2vec是学习词向量表示的算法框架  
(Mikolov et al., 2013) ;
- 基本思想：
  - 给定一段长文本；
  - 文本中的每个词汇表示为一个（连续值）向量；
  - 用一个滑动窗口  $t$  遍历文本，窗口包含一个中心词  $c$  及其上下文词  $o$ ；
  - 用  $c$  和  $o$  的词向量相似度**计算给定  $c$  的情况下  
○出现的概率**（反之亦然）；
  - 持续调整词向量，以最大化这个概率。



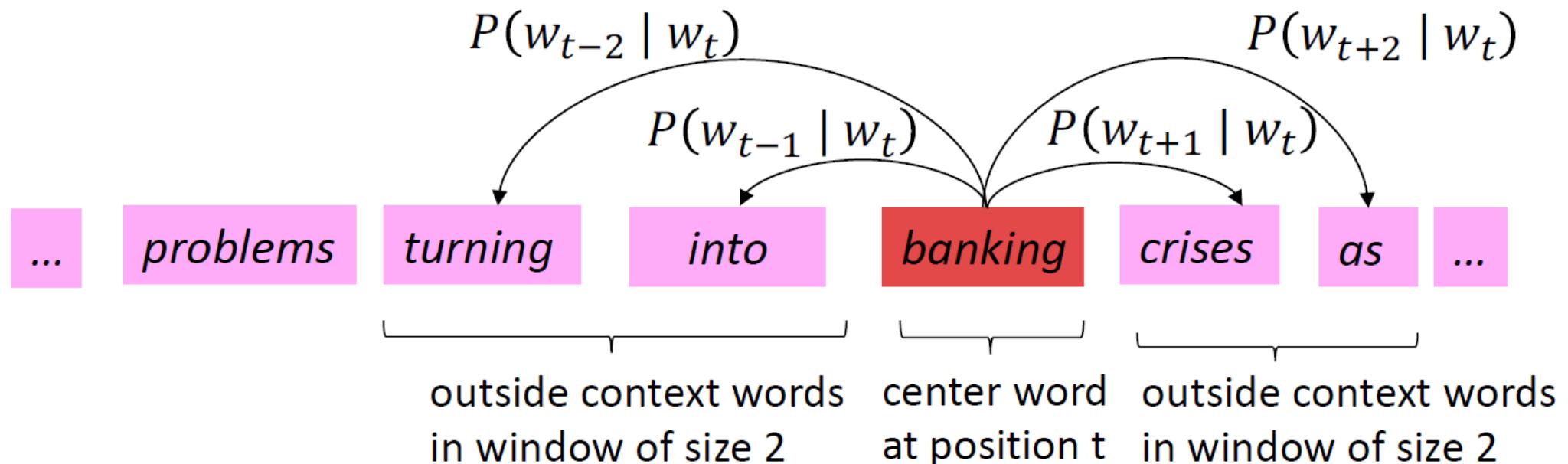
# Word2vec

- Word2vec



# Word2vec

- Word2vec



# Word2vec

## • 目标函数

- 对于每个位置  $t = 1, \dots, T$ , 中心词为  $w_t$ , 预测大小为  $m$  的窗口的上下文词出现的概率;
- 似然函数:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^T \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} | w_t; \theta)$$

$\theta$  is all variables to be optimized

- 优化的目标函数是**平均负对数似然**:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

- 优化目标: 最小化目标函数值 = 最大化预测准确率。

# Word2vec

## • 目标函数

- 优化的目标函数是**平均负对数似然**:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

- 如何计算  $P(w_{t+j} | w_t; \theta)$ ?

- 每个词  $w$  使用两个向量来表示:

$v_w$ : 当  $w$  为**中心词**;

$u_w$ : 当  $w$  为**上下文词**;

- 对于一个中心词  $c$  和一个上下文词  $o$ :  $P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$  **(Softmax)**

保证为正数      利用点积表示  $o$  和  $c$  的相似度

$$\exp(u_o^T v_c)$$

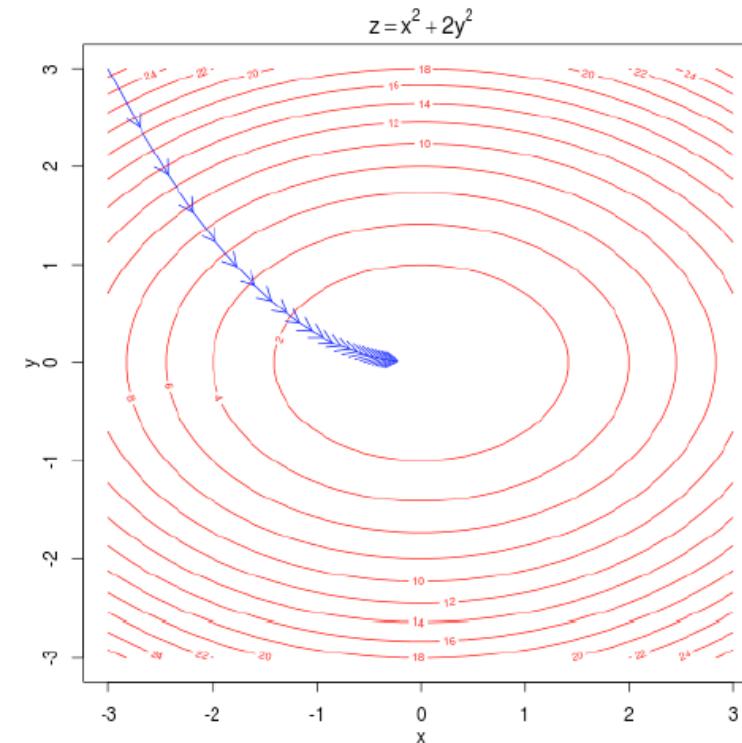
归一化

# Word2vec

## ● 模型训练

- 通过梯度下降法逐步调整参数，以最小化损失函数；
- $\theta$  表示模型的所有参数；
- 假设词向量维度为  $d$ ，词的数量为  $V$ ，由于每个词使用2个向量表示， $\theta$  的长度为  $2dV$ 。

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$



# Word2vec

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

$U$

outside

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

$V$

center

$$\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}$$

$U \cdot v_4^T$

dot product

$$\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}$$

$\text{softmax}(U \cdot v_4^T)$

probabilities

# Word2vec

## ● 模型细节

- Word2vec有两种常见的形式：

- 连续词袋模型 (Continuous bag of words, CBOW)
- 跳字模型 (Skip-gram)

- **连续词袋模型 (CBOW) :**

- 给定上下文词，预测中心词；
- 训练目标：最大化预测中心词的概率——得到表示上下文语境的词向量；

- **跳字模型 (Skip-gram) :**

- 给定中心词，预测上下文词；
- 训练目标：最大化预测上下文词的概率——得到表示词语语义的词向量。

# Word2vec

## ● 模型细节

- 条件概率计算:  $P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$
- 每步梯度计算都包含词典大小的 (~几十万) 的项的累加, 计算开销很大;
- 简化计算——**负采样 (Negative sampling)** :
  - 思想: 训练一个**二分类模型**, 用于区分真正同时出现的词和“随机”组合在一起的词(噪声);
  - 方法: 取K个负样本, 最大化真实共现的概率, 最小化随机组合的概率;
  - 修改了原来的目标函数:

$$J_{neg\text{-}sample}(\mathbf{u}_o, \mathbf{v}_c, U) = - \log \sigma(\mathbf{u}_o^T \mathbf{v}_c) - \sum_{k \in \{K \text{ sampled indices}\}} \log \sigma(-\mathbf{u}_k^T \mathbf{v}_c)$$

  
**sigmoid rather than softmax**

# Word2vec

## ● 优点

- 词向量维度低，方便存储、处理，且通用性强；
- 能够很好地计算词与词之间的相似度。

```
In [10]: result = model.most_similar(u"青蛙")
```

```
In [11]: for e in result:  
    print e[0], e[1]  
....:
```

```
老鼠 0.559612870216  
乌龟 0.489831030369  
蜥蜴 0.478990525007  
猫 0.46728849411  
鳄鱼 0.461885392666  
蟾蜍 0.448014199734  
猴子 0.436584025621  
白雪公主 0.434905380011  
蚯蚓 0.433413207531  
螃蟹 0.4314712286
```

```
In [20]: result = model.most_similar(u"清华大学")
```

```
In [21]: for e in result:  
    print e[0], e[1]
```

```
....:  
北京大学 0.763922810555  
化学系 0.724210739136  
物理系 0.694550514221  
数学系 0.684280991554  
中山大学 0.677202701569  
复旦 0.657914161682  
师范大学 0.656435549259  
哲学系 0.654701948166  
生物系 0.654403865337  
中文系 0.653147578239
```

## • 缺点

- 词和向量是一一对应关系，无法处理一词多义的问题；
- 是一种静态的方式，无法针对特定任务做动态优化。

# Word2vec

The screenshot shows the Gensim documentation website for the Word2Vec model. The top navigation bar includes links for Home, Documentation, Support, API, About, and Donate. A purple sidebar on the left contains version information (4.3.0), a search bar, and links to 'What is Gensim?' and various documentation sections like Core Tutorials, How-to Guides, Other Resources, and Tutorials: Learning Oriented Lessons. The main content area features a red-bordered box asking for sponsorship. Below it, the breadcrumb navigation shows the path: Home > Documentation > Tutorials: Learning Oriented Lessons > Word2Vec Model. A blue note box provides a download link for example code. The main article title is 'Word2Vec Model', which introduces the model and its use on the Lee Evaluation Corpus. A code snippet in a green box shows basic logging setup. The final paragraph explains what Word2Vec is and how it works.

4.3.0

Search docs

What is Gensim?

Documentation

- Core Tutorials: New Users Start Here!
- Tutorials: Learning Oriented Lessons
- How-to Guides: Solve a Problem

Other Resources

- Core Tutorials: New Users Start Here!

Tutorials: Learning Oriented Lessons

- Word2Vec Model
- Doc2Vec Model
- Ensemble LDA
- FastText Model
- Fast Similarity Queries with Annoy and Word2Vec
- LDA Model
- Word Mover's Distance

Please sponsor Gensim to help sustain this open source project!

Home > Documentation > Tutorials: Learning Oriented Lessons > Word2Vec Model

**Note**

Click [here](#) to download the full example code

## Word2Vec Model

Introduces Gensim's Word2Vec model and demonstrates its use on the [Lee Evaluation Corpus](#).

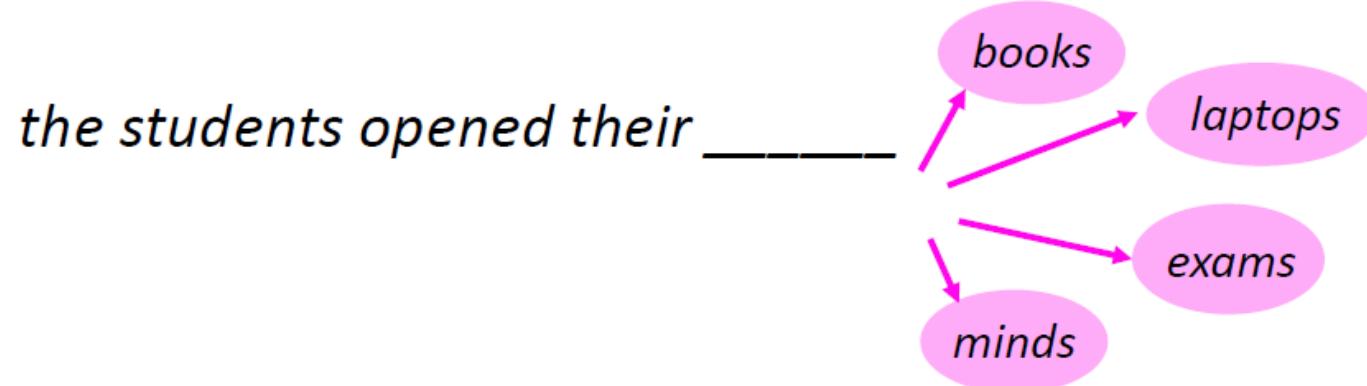
```
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
```

In case you missed the buzz, Word2Vec is a widely used algorithm based on neural networks, commonly referred to as "deep learning" (though word2vec itself is rather shallow). Using large amounts of unannotated plain text, word2vec learns relationships between words automatically. The output are vectors, one vector per word, with remarkable linear relationships that allow us to do things like:

# 语言模型

- 语言模型 (Language model)

- 基本任务：预测“下一个词”



- 给定一个单词序列  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(t)}$ ，计算下一个词  $\mathbf{x}^{(t+1)}$  的概率分布：

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})$$

其中， $\mathbf{x}^{(t+1)}$  是词典  $V = \{\mathbf{w}_1, \dots, \mathbf{w}_{|V|}\}$  中的任意单词。

## • 语言模型 (Language model)

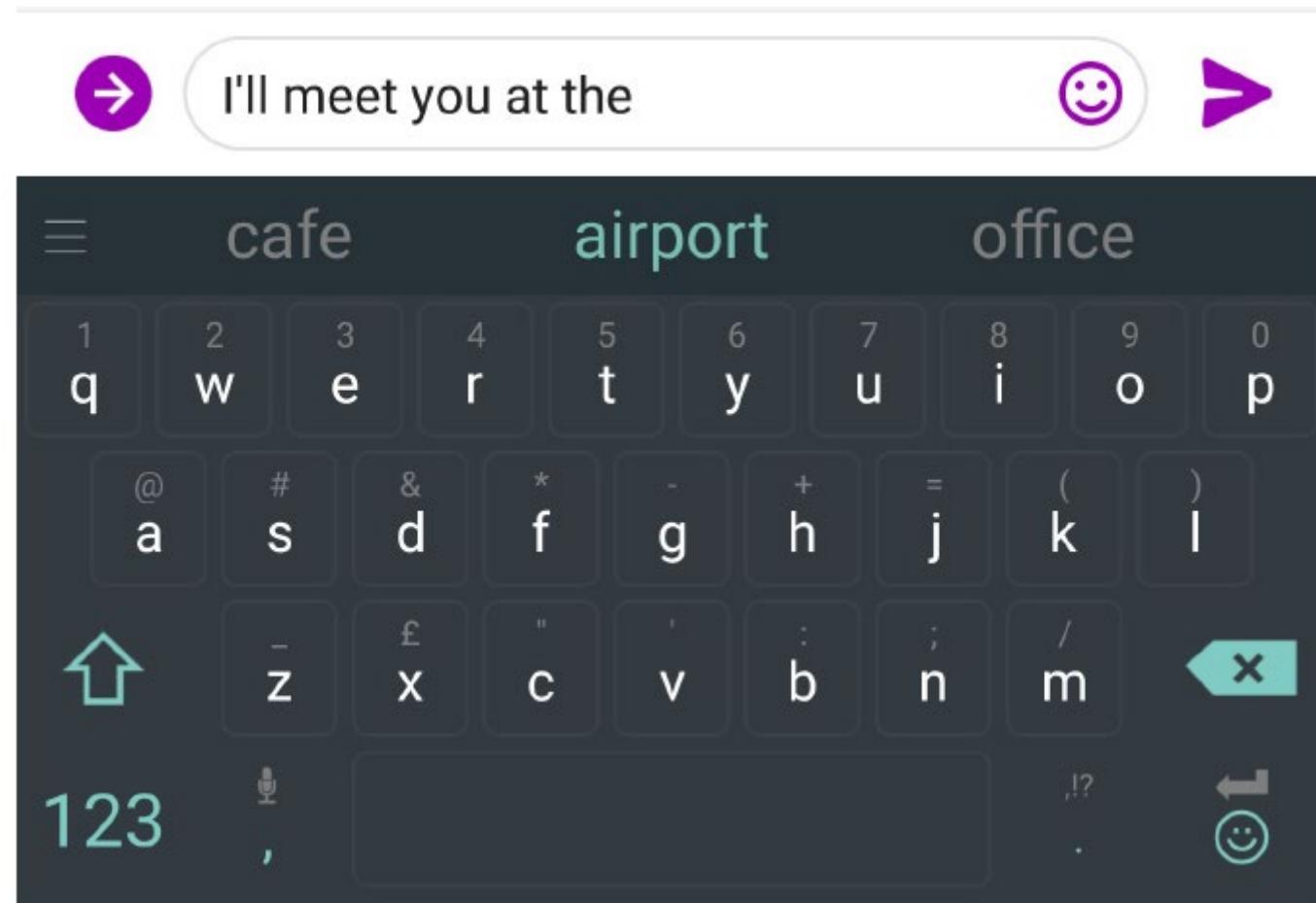
- 从另一个视角，语言模型为某个特定语段的出现计算了概率；
- 例如，对于某个语段  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$ ，其出现的概率可按以下方式计算：

$$\begin{aligned} P(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}) &= P(\mathbf{x}^{(1)}) \times P(\mathbf{x}^{(2)} | \mathbf{x}^{(1)}) \times \cdots \times P(\mathbf{x}^{(T)} | \mathbf{x}^{(T-1)}, \dots, \mathbf{x}^{(1)}) \\ &= \prod_{t=1}^T P(\mathbf{x}^{(t)} | \underbrace{\mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}}_{}) \end{aligned}$$

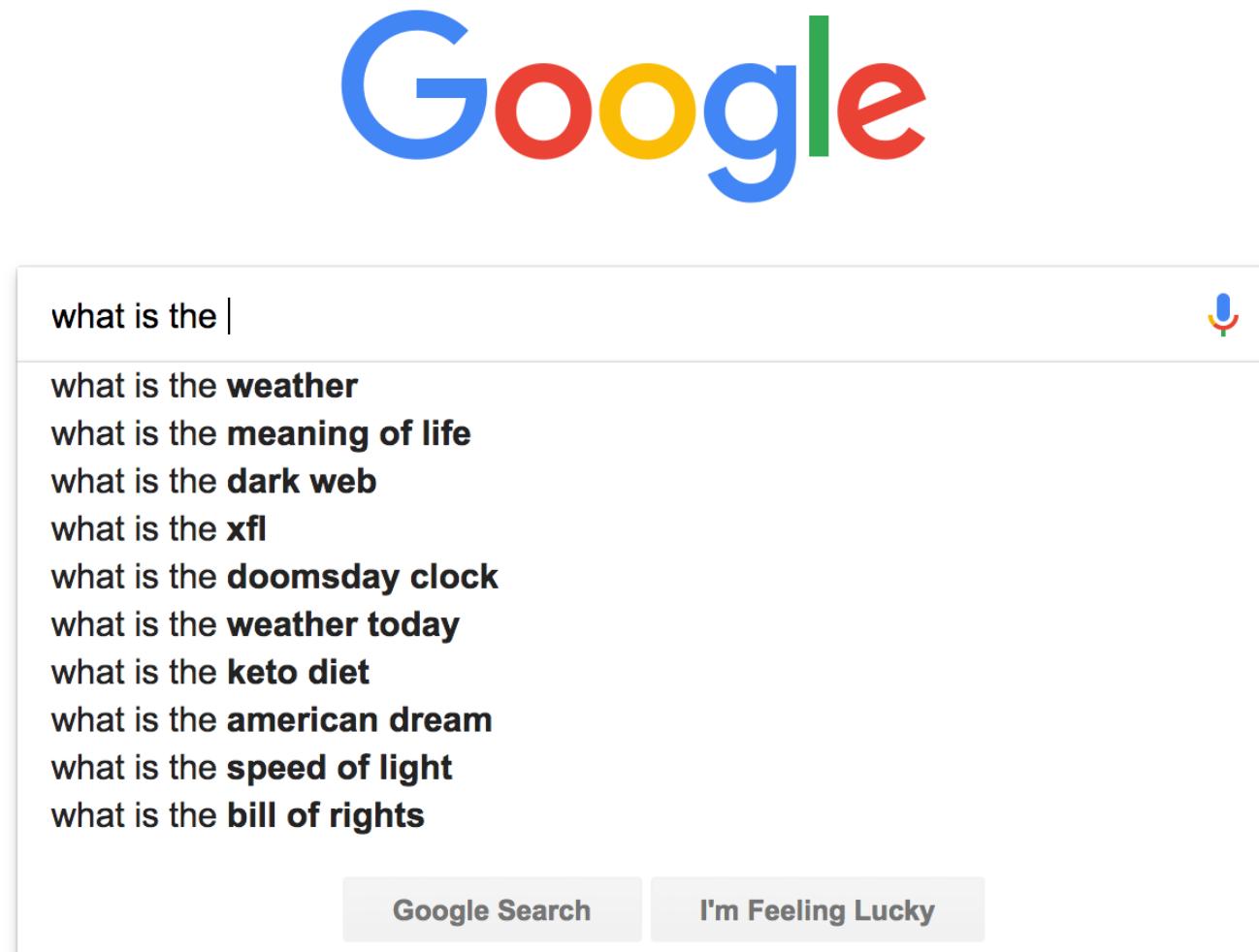
由语言模型计算得到

# 语言模型

- 语言模型 (Language model)



- 语言模型 (Language model)



# 语言模型

## ● n-gram语言模型

- 在计算语言学中，n-gram是指文本中连续的n个item（例如n个词）；

*the students opened their \_\_\_\_\_*

- **unigrams**: “the”, “students”, “opened”, “their”
  - **bigrams**: “the students”, “students opened”, “opened their”
  - **trigrams**: “the students opened”, “students opened their”
  - **four-grams**: “the students opened their”
- 
- n-gram语言模型：计算不同n-gram出现的频率，用于预测“下一个词”。

# 语言模型

## • n-gram语言模型

- 马尔科夫假设:  $\mathbf{x}^{(t+1)}$  仅依赖其前面出现的  $n-1$  个词;

$$\begin{aligned} P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)}) &= P(\mathbf{x}^{(t+1)} | \overbrace{\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)}}^{n-1 \text{ words}}) \\ &= \frac{P(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})}{P(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})} \end{aligned}$$

**n-gram**  
**(n-1)-gram**

- 根据大数定律, 我们统计频数以得到概率的近似:

$$\frac{P(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})}{P(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})} \approx \frac{\text{count}(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})}{\text{count}(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})}$$

# 语言模型

## • n-gram语言模型

- 案例：假如我们试图建立一个4-gram语言模型

*as the proctor started the clock, the students opened their \_\_\_\_\_*

舍弃  仅考虑前面3个词

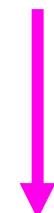
$$P(\mathbf{w}|\text{students opened their}) = \frac{\text{count(students opened their } \mathbf{w})}{\text{count(students opened their)}}$$

- 假设语料库中，`students opened their` 共出现了1000次；
- `students opened their books`出现了400次：  $P(\text{books} \mid \text{students opened their}) = 0.4$
- `students opened their exams`出现了100次：  $P(\text{exams} \mid \text{students opened their}) = 0.1$

# 语言模型

- 利用n-gram语言模型进行文本生成

Today the \_\_\_\_\_



得到概率分布，即前2个词是 today the 的条件下，接下来出现某个词的概率

company 0.153

bank 0.153

price 0.077

italian 0.039

emirate 0.039

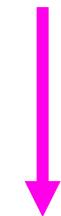
...

例如， price出现的概率为0.077

# 语言模型

- 利用n-gram语言模型进行文本生成

Today the price \_\_\_\_\_



得到概率分布，即前2个词是 the price 的条件下，接下来出现某个词的概率

of	0.308
for	0.050
it	0.046
to	0.046
is	0.031
...	

例如，of 出现的概率为 0.308

# 语言模型

- 利用n-gram语言模型进行文本生成

Today the price of \_\_\_\_\_

得到概率分布，即前2个词是 price of 的条件下，接下来出现某个词的概率

the	0.072
18	0.043
oil	0.043
its	0.036
gold	0.018
...	

例如，gold 出现的概率为 0.018

# 语言模型

- 利用n-gram语言模型进行文本生成

*today the price of gold per ton , while production of shoe  
lasts and shoe industry , the bank intervened just after it  
considered and rejected an imf demand to rebuild depleted  
european stocks , sept 30 end primary 76 cts a share .*

# 语言模型

- n-gram语言模型的不足

*the students opened their \_\_\_\_\_*

问题1：students opened their  $w$ 可能从未在语料库出现过

$$P(w| \text{students opened their}) = \frac{\text{count(students opened their } w\text{)}}{\text{count(students opened their)}}$$

问题2：students opened their 也可能从未在语料库出现过

- 稀疏性问题；
- 对于n-gram模型，随着n的增大，模型的稀疏性也随之增长。

# 语言模型

- 问题：如何用神经网络构建语言模型？

## 固定窗口神经语言模型

输出分布

$$\hat{y} = \text{softmax}(\mathbf{U}\mathbf{h} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

Hidden layer

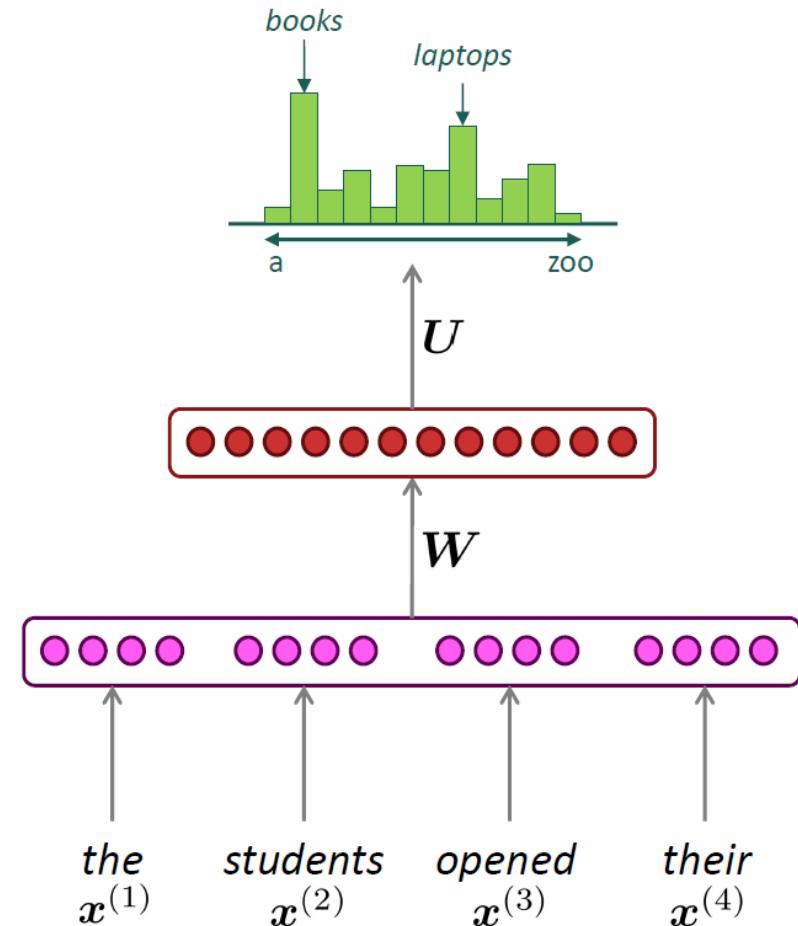
$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}_1)$$

词嵌入

$$\mathbf{e} = [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \mathbf{e}^{(3)}; \mathbf{e}^{(4)}]$$

单词

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$$



# 语言模型

- 问题：如何用神经网络构建语言模型？

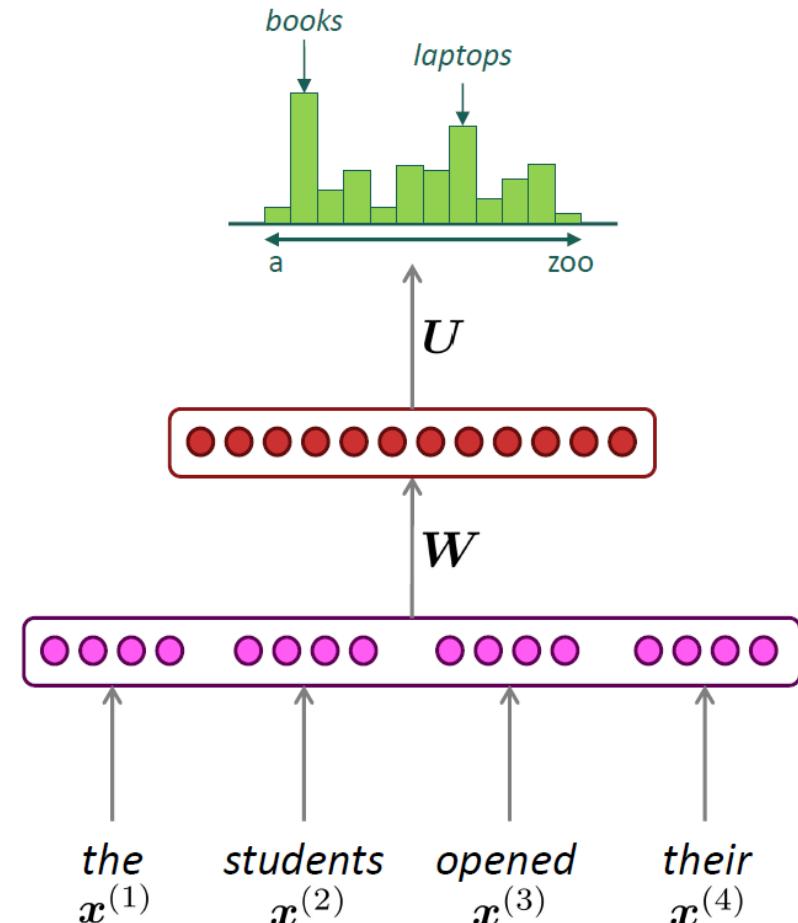
## 固定窗口神经语言模型

优点：

- 与n-gram语言模型相比，解决了稀疏问题；

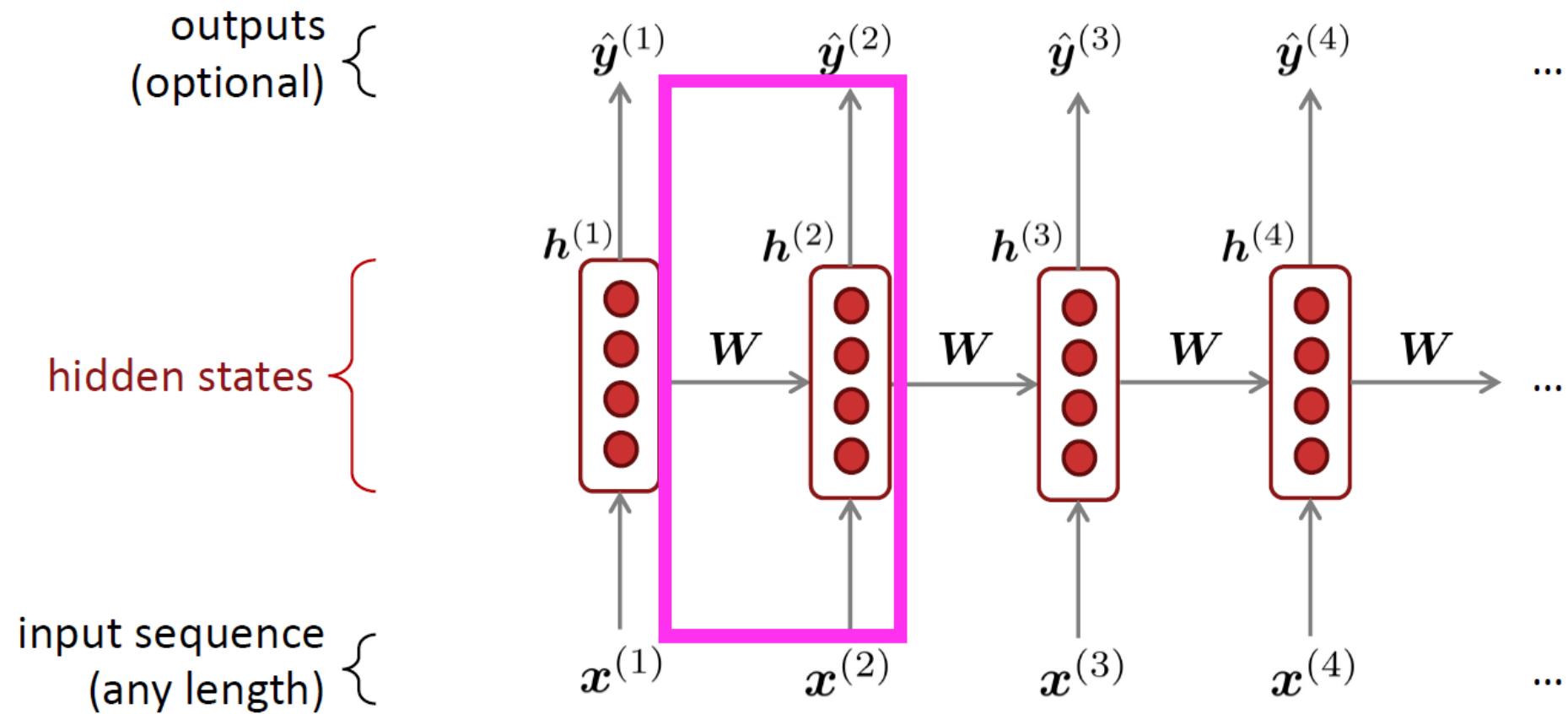
不足：

- 需要固定窗口大小，如果增大窗口，将导致权重矩阵W规模增大；
- 不存在对称性， $x^{(1)}$ 、 $x^{(2)}$ 交换位置将导致完全不同的结果。



# 语言模型

- 基于RNN构建语言模型



# 语言模型

## ● 基于RNN构建语言模型

输出分布

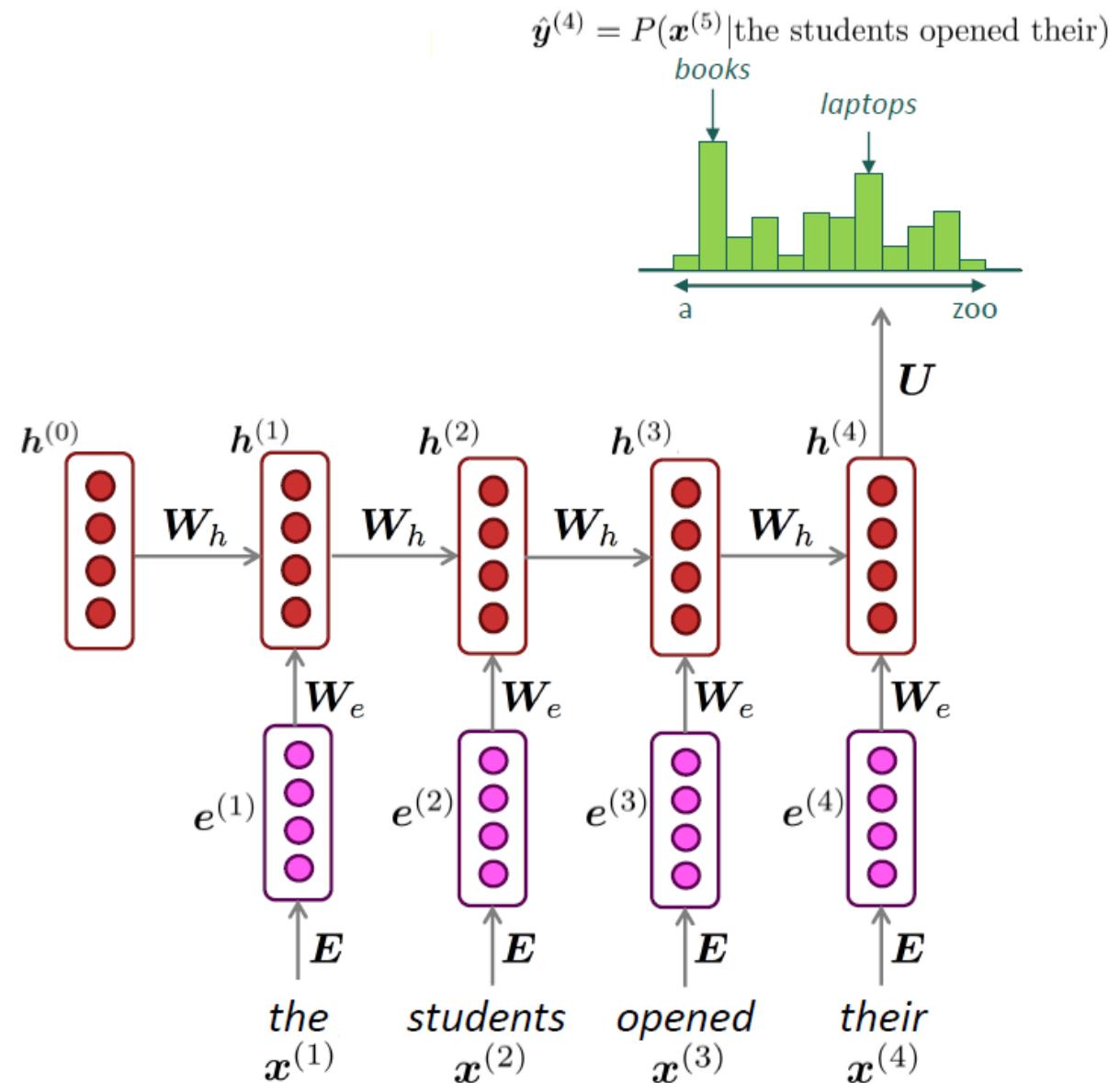
$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}\mathbf{h}^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

Hidden layer

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_e \mathbf{e}^{(t)} + \mathbf{b}_1)$$

词嵌入  $\mathbf{e}^{(t)} = \mathbf{E}\mathbf{x}^{(t)}$

单词  $\mathbf{x}^{(t)} \in \mathbb{R}^{|V|}$



# 语言模型

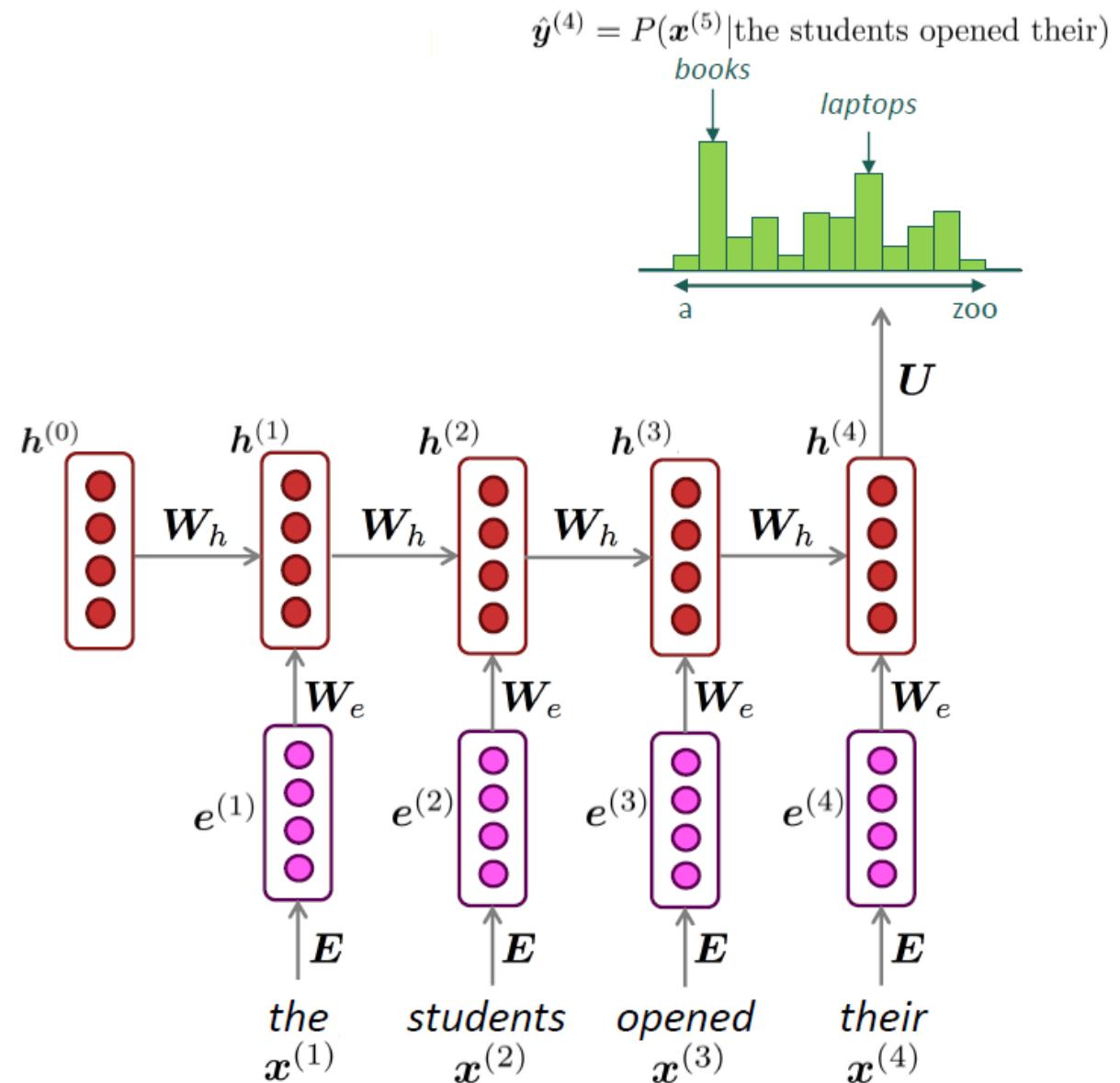
## ● 基于RNN构建语言模型

**优点：**

- 文本长度不受限制；
- 随着文本长度的增长，由于共享权重的机制，模型大小不会增长

**不足：**

- RNN存在长程依赖问题；
- 训练、推理较慢。



# 语言模型

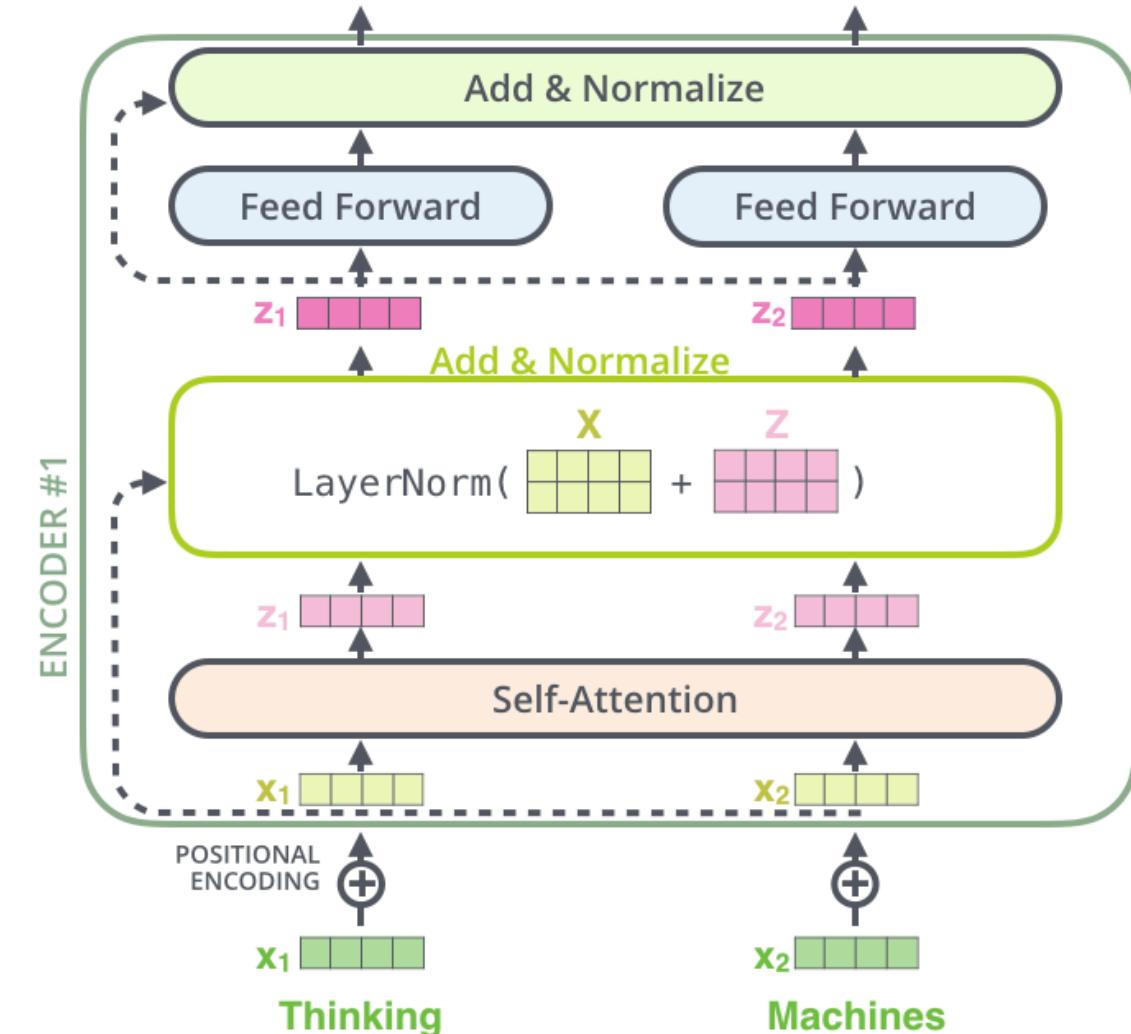
- 基于Transformer构建语言模型

## 优点:

- 文本长度不受限制;
- 比RNN更好地上下文关联能力;

## 不足:

- 训练、推理较慢，消耗计算资源多。



# Seq2Seq与机器翻译

## ● 机器翻译 (Machine Translation, MT)

- 任务: 将源语言 (Source language) 语段 x 翻译为目标语言 (Target language) 语段 y;

x: *L'homme est né libre, et partout il est dans les fers*

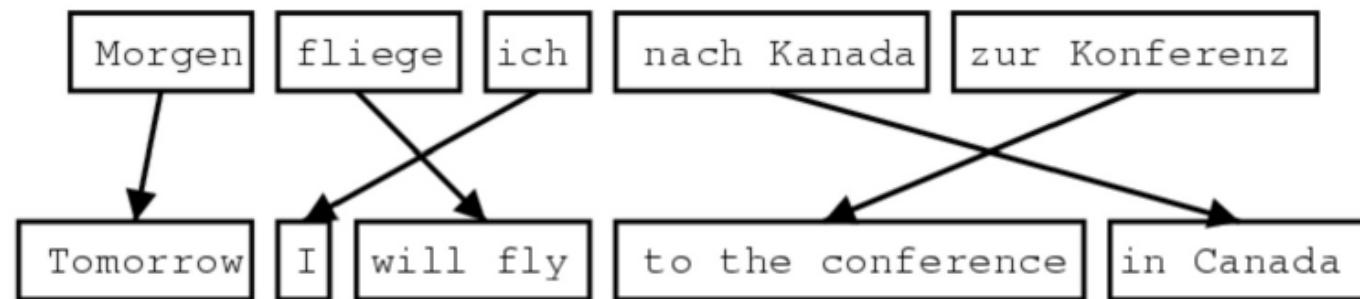


y: *Man is born free, but everywhere he is in chains*

## ● 早期的机器翻译

- 机器翻译相关研究最早出现在20世纪50年代；
- 军事、情报部门投入大量资源进行机器翻译的研究；
- 主要是进行不同语言之间的单词替换，效果很差。

# 机器翻译



1519年600名西班牙人在墨西哥登陆，去征服几百万人口的阿兹特克帝国，初次交锋他们损兵三分之二。

In 1519, six hundred Spaniards landed in Mexico to conquer the Aztec Empire with a population of a few million. They lost two thirds of their soldiers in the first clash.

translate.google.com (2009): 1519 600 Spaniards landed in Mexico, millions of people to conquer the Aztec empire, the first two-thirds of soldiers against their loss.

translate.google.com (2013): 1519 600 Spaniards landed in Mexico to conquer the Aztec empire, hundreds of millions of people, the initial confrontation loss of soldiers two-thirds.

translate.google.com (2015): 1519 600 Spaniards landed in Mexico, millions of people to conquer the Aztec empire, the first two-thirds of the loss of soldiers they clash.

## • 统计机器翻译 (1990s~2010s)

- 源语言句子  $x$ ，目标语言句子  $y$ ；
- 统计机器翻译建立了以下目标：

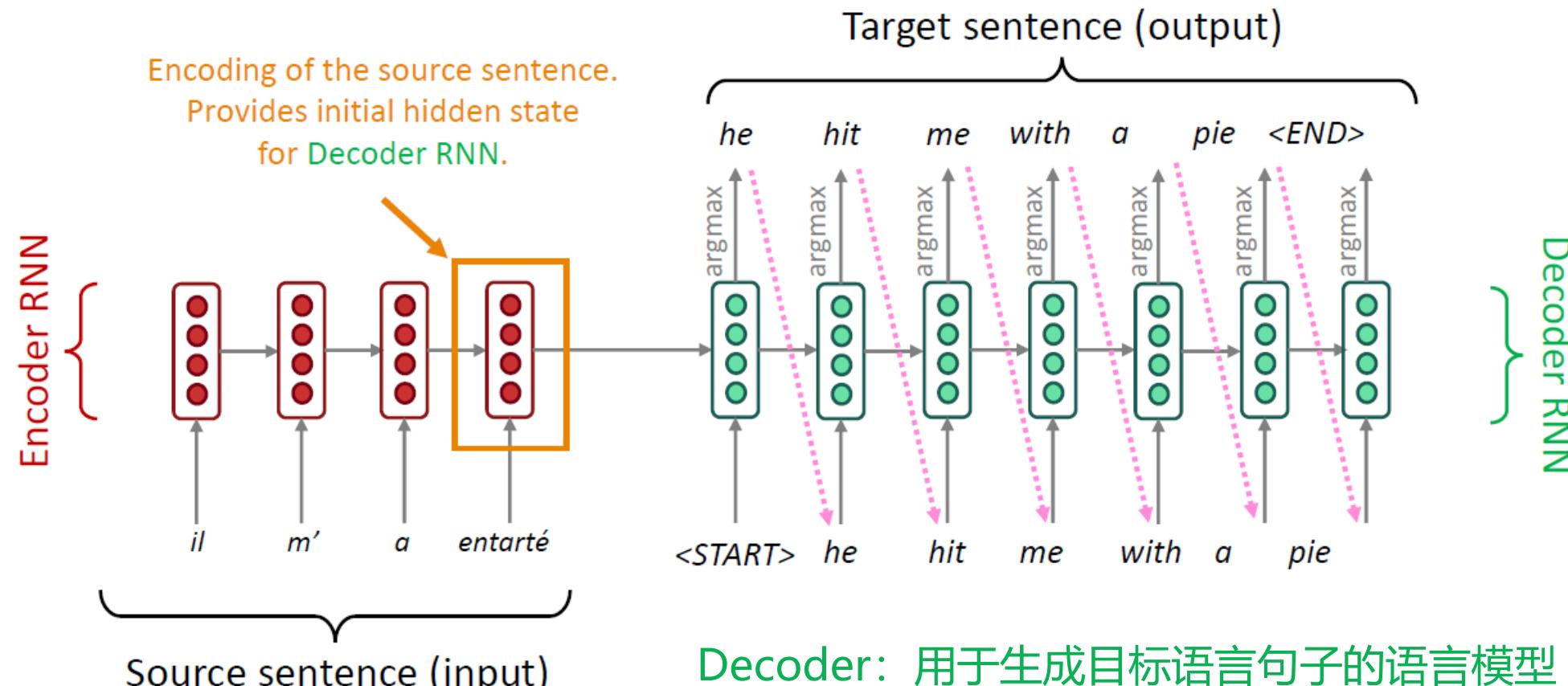
$$\operatorname{argmax}_y P(y|x)$$

- 根据贝叶斯定理，上式可以改写为：

$$\operatorname{argmax}_y P(x|y)P(y)$$

- $P(x|y)$ : 翻译模型——通过多语言平行数据学习；
- $P(y)$ : 语言模型——通过某种特定语言的数据进行学习。

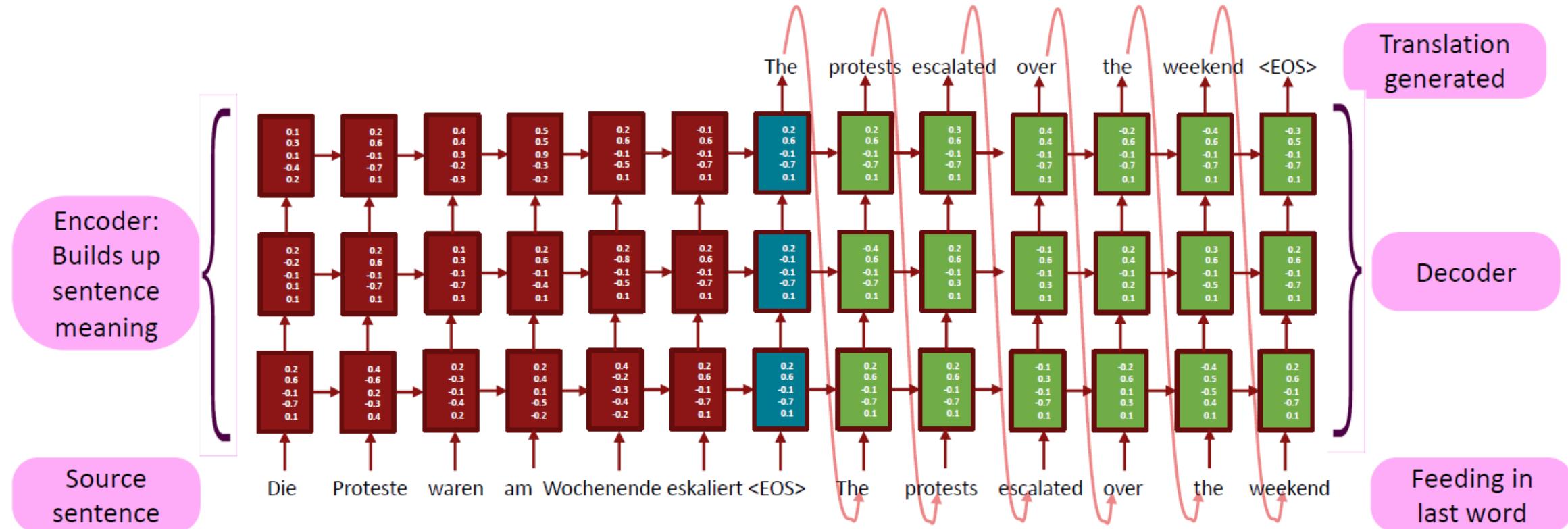
## • 基于神经网络的机器翻译——Seq2seq模型



Encoder: 将源语言句子编码为隐层特征

# 机器翻译

- 基于神经网络的机器翻译——Seq2seq模型



# 大规模语言模型

# 大规模语言模型

## ● 产生背景：

- 计算机视觉领域采用ImageNet数据集对模型进行预训练，使得模型可以通过海量图像充分学习如何提取特征，然后再根据任务目标进行模型精调；
- 受此影响，自然语言处理领域基于预训练语言模型的方法也逐渐成为主流；
- 以GPT为代表的基于Transformer的大规模预训练语言模型的出现，使得自然语言处理全面进入了**预训练微调范式**新时代。

## ● 优势：

- 将预训练模型应用于下游任务时，不需要了解太多的任务细节，不需要设计特定的神经网络结构，只需要“**微调**”（**finetune**）预训练模型，即使用具体任务的标注数据在预训练语言模型上进行监督训练，就可以取得显著的性能提升。

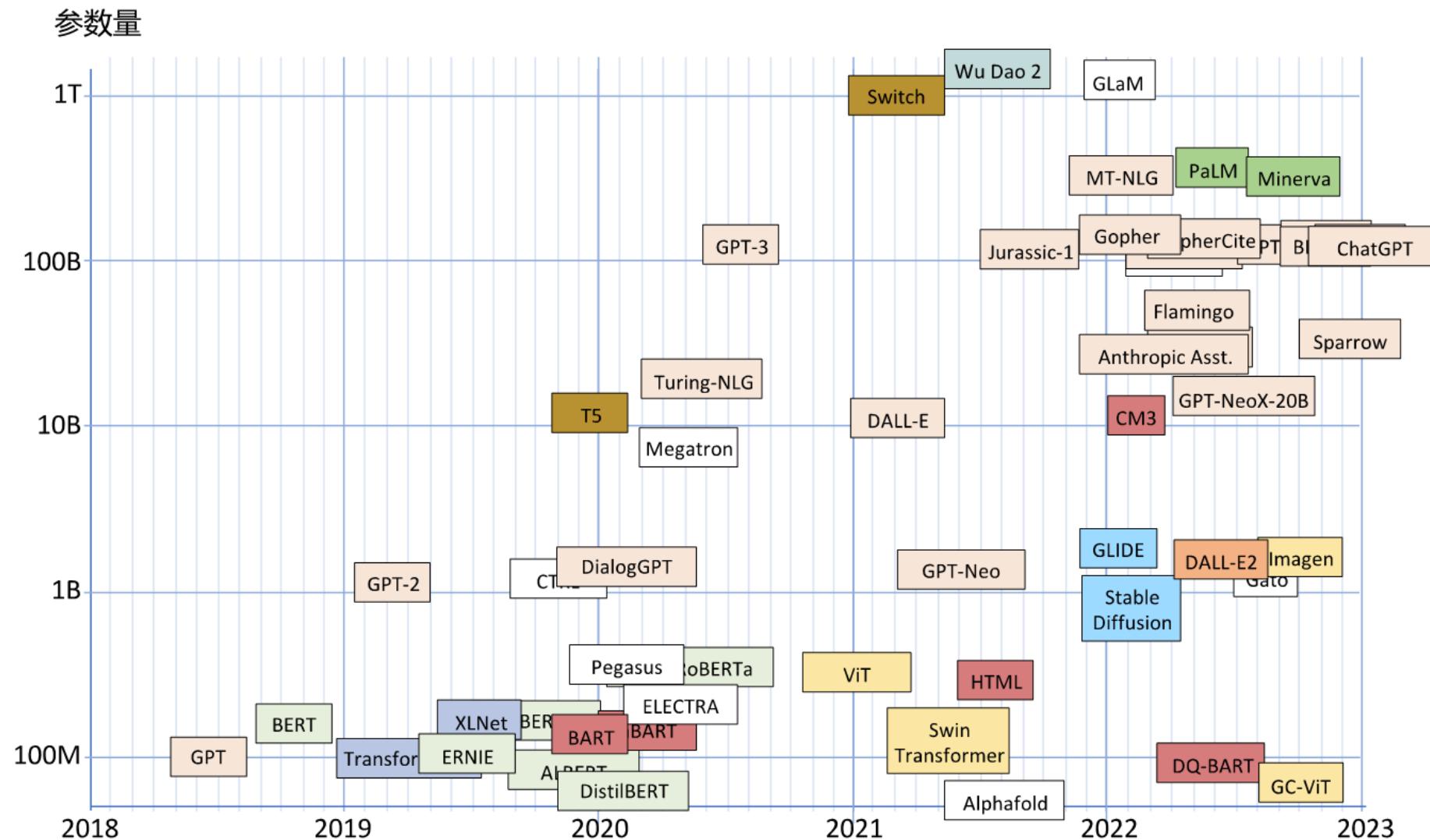
# 大规模语言模型

- “百模大战”：

- 2020年，Open AI发布了包含1750亿参数的生成式大规模预训练语言模型GPT 3 (Generative Pre-trained Transformer 3)；
- 此后，Google、Meta、百度等公司和研究机构都纷纷发布了PaLM、LaMDA、T0等不同大规模语言模型 (Large Language Model, LLM)，也称大模型。

模型名称	参数量	训练单词数	研发机构	发布时间
ChatGPT	1750 亿	3000 亿	OpenAI	2022 年 11 月
Galactica	1200 亿	4500 亿	Meta AI	2022 年 11 月
BLOOMZ	1760 亿	3660 亿	BigScience	2022 年 11 月
U-PaLM	5400 亿	7800 亿	Google Research	2022 年 10 月
CodeGeeX	130 亿	8500 亿	清华大学	2022 年 9 月
PaLM	5400 亿	7800 亿	Google Research	2022 年 4 月
ERNIE 3.0 Titan	2600 亿	-	Baidu	2021 年 12 月
FLAN	1370 亿	-	Google	2021 年 9 月
GPT-3	1750 亿	3000 亿	OpenAI	2020 年 5 月
T5	110 亿	340 亿	Google	2019 年 10 月
RoBERTa	3.55 亿	22000 亿	Meta AI	2019 年 7 月
GPT-2	15 亿	100 亿	OpenAI	2019 年 2 月
BERT	3 亿	1370 亿	Google	2018 年 10 月
GPT-1	1 亿	-	OpenAI	2018 年 6 月

# 大规模语言模型



# 大模型训练

- 整体过程可以分为三个阶段：

- 第一个阶段是**基础大模型训练**，该阶段主要完成长距离语言模型的预训练，通过代码预训练使得模型具备代码生成的能力；
- 第二阶段是**指令微调（Instruct Tuning）**，通过给定指令进行微调的方式使得模型具备完成各类任务的能力；
- 第三个阶段是**类人对齐**，加入更多人工提示词，并利用有监督微调并结合基于强化学习的方式，使得模型输出更贴合人类需求。

# 大模型训练

## ● 基础大模型训练：

- 训练数据集主要包含经过过滤的Common Crawl数据集、WebText2、Books1、Books2以及英文 Wikipedia等数据集；
- 其中Common Crawl的原始数据有45TB，进行过滤后保留了570GB的数据；
- 通过子词方式对语料进行切分，大约一共包含5000亿子词；
- 为了保证模型使用更多高质量数据进行训练，在GPT-3训练时，根据语料来源的不同，设置不同的采样权重，在完成3000亿子词训练时，英文Wikipedia的语料平均训练轮数为3.4次，而Common Crawl和Books 2为0.44次和0.43次；
- 据估计，OpenAI使用了约1万块NVIDIA A100 GPU来训练GPT-3.5模型，后来为了进一步满足服务器需求，将GPU数量提升到了3万块以上。



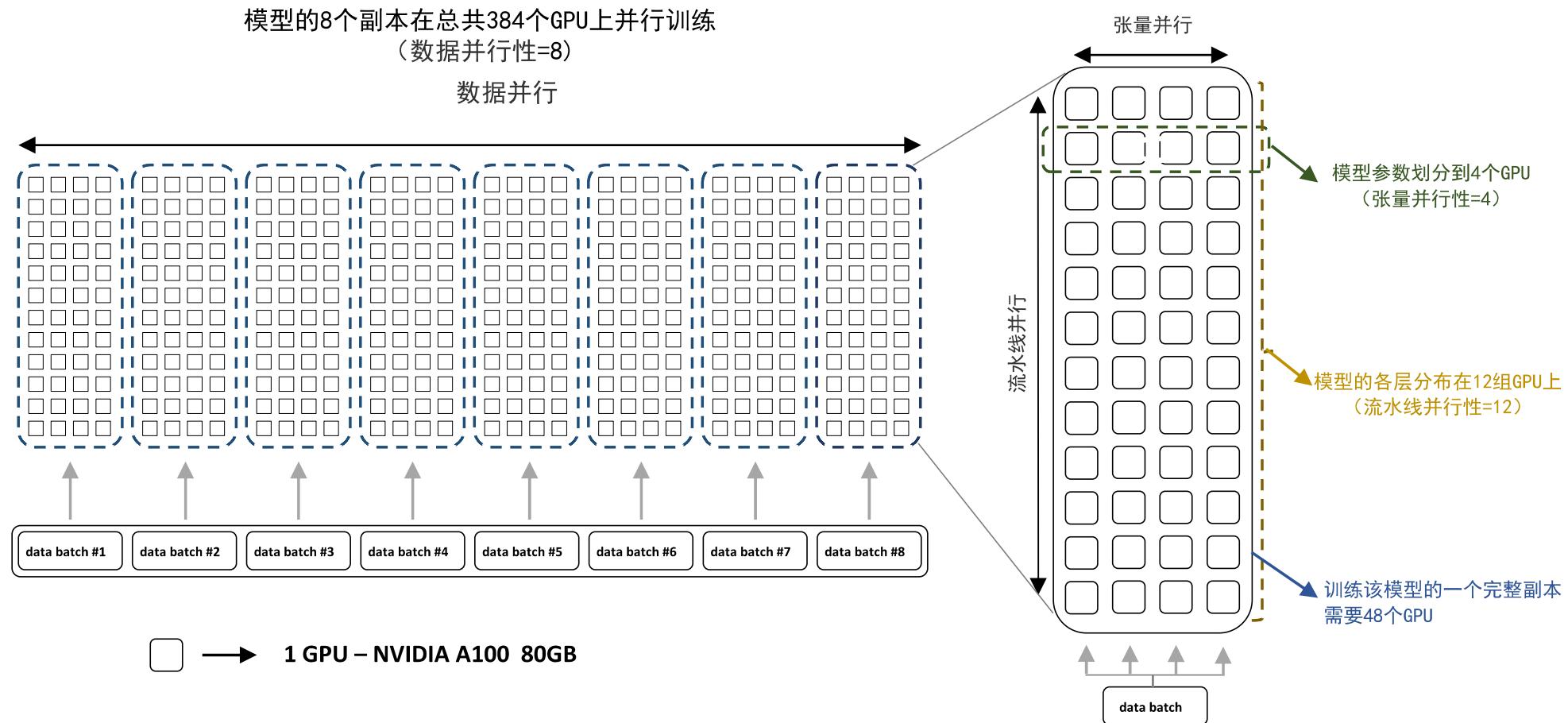
Common Crawl  
maintains a **free, open**  
**repository** of web crawl  
data that can be used by  
anyone.



# 大模型训练

## ● 基础大模型训练：

- BLOOM (与GPT3类似的模型) 使用Megatron-DeepSpeed分布式训练框架



# 大模型训练

## • 指令微调：

- 预训练语言模型可以根据任务数据进行微调（Fine-tuning），这种范式能够应用于中等规模参数量（几百万到几亿规模）的预训练模型，但是针对成百上千亿参数规模的大模型，针对每个任务都进行微调的计算开销和时间成本几乎都是不可接受的；
- 因此，研究员们提出了指令微调（Instruction Finetuning）方案，将大量各类型任务，统一为生成式自然语言理解框架，并构造训练语料进行微调。

例如，可以将情感倾向分析任务，通过如下指令，将贬义和褒义的分类问题转换到生成式自然语言理解框架：

For each snippet of text, label the sentiment of the text as positive or negative.

Text: this film seems thirsty for reflection, itself taking on adolescent qualities.

Label: [positive / negative]

# 大模型训练

指令  
不包含样例

指令  
包含样例

训练目标不包含思维链

Answer the following yes/no question.

Can you write a whole Haiku in a single tweet?



yes

Answer the following yes/no question by reasoning step-by-step.

Can you write a whole Haiku in a single tweet?



A haiku is a Japanese three-line poem. That is short enough to fit in 280 characters. The answer is yes.

Q: Answer the following yes/no question.

Could a dandelion suffer from hepatitis?

A: no



yes

Q: Answer the following yes/no question.

Can you write a whole Haiku in a single tweet?

A:

Q: Answer the following yes/no question by reasoning step-by-step.

Could a dandelion suffer from hepatitis?

A: Hepatitis only affects organisms with livers.

Dandelions don't have a liver. The answer is no.



A haiku is a Japanese three-line poem. That is short enough to fit in 280 characters. The answer is yes.

Q: Answer the following yes/no question by reasoning step-by-step.

Can you write a whole Haiku in a single tweet?

A:

# 大模型训练

## • 类人对齐：

- 经过指令微调后的模型，虽然在开放领域任务能力表现优异，但是模型输出的结果通常是简单答案，与人类的回答相差很大，因此需要进一步优化模型，使其可以生成更加贴近人类习惯的文本内容；
- 自然语言处理语料集合中所包含的答案，通常都是简短的标签，基本没有类似人类回答的有监督数据，如果将自然语言处理任务数据集进行人工改造，所需要的时间成本和人工成本过于高昂；
- Open AI提出了使用**基于人类反馈的强化学习方法 (Reinforcement Learning from Human Feedback, RLHF)**，从而大幅度降低了数据集构建成本，并达到了非常好的效果。

# 大模型训练

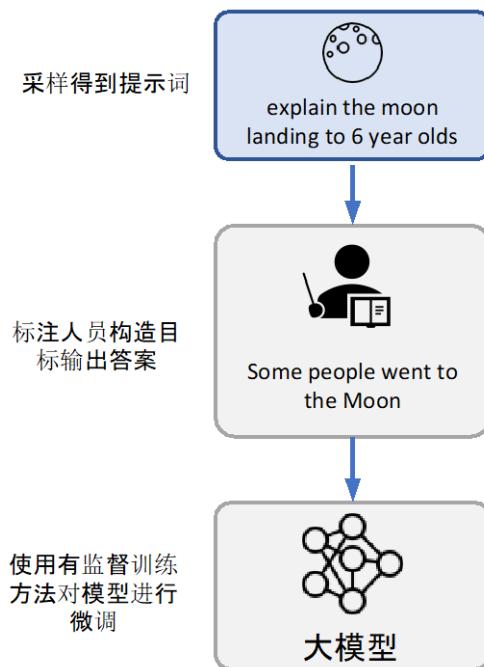
## • 类人对齐：

- RLHF算法主要分为以下三个步骤：
  - **Step1. 有监督微调**：人类标注者准备一组提示和期望结果，对预训练模型进行有监督微调；
  - **Step2. 创建奖励模型**：人类标注者利用模型生成一组对比数据，也就是几对指令、答案，并根据偏好对它们进行排序，根据这些排序数据训练奖励模型，以便在没有人为干预的情况下自动对强化学习智能体的输出进行排序；
  - **Step3. 使用奖励模型训练 RL 策略**：创建一个反馈环来训练和微调强化学习策略，利用奖励模型生成最优回答。这个过程会迭代进行，直到达到预期的性能水平。

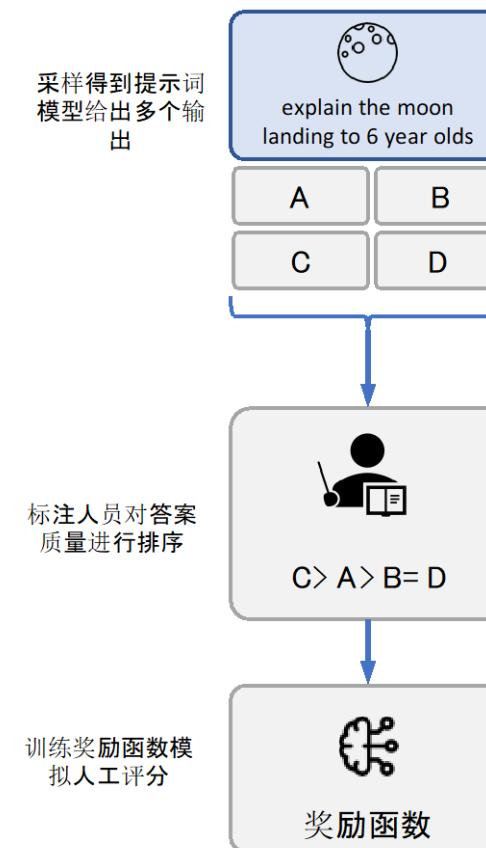
# 大模型训练

## • 类人对齐：

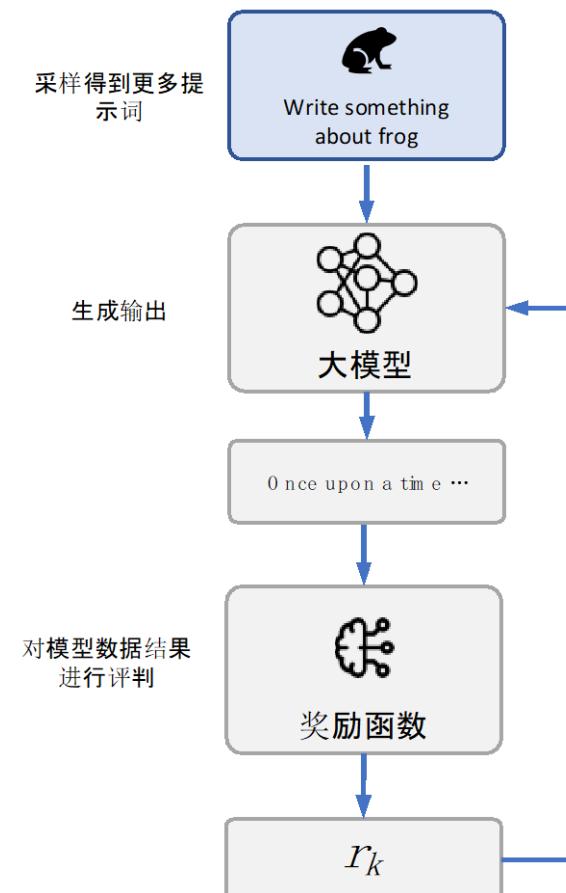
步骤1：收集初始数据  
初步训练大模型



步骤2：收集打分数据  
并训练奖励函数



步骤3：利用强化学习机制根据奖励得分进一步优化大模型



# 提示工程

2024/4/28 18:39:29



解释一下机器学习是什么。



2024/4/28 18:39:44

机器学习是人工智能的一个分支，旨在让计算机系统通过学习经验和数据，而不是通过显式编程来完成特定任务。它涉及构建和训练模型，使计算机能够根据数据进行预测、决策、分类或者识别模式。

基本上，机器学习算法使计算机能够自动学习并改进其性能，而无需明确地编程。这是通过提供大量数据来训练模型，并根据模型的输出进行调整来实现的。

机器学习分为监督学习、无监督学习和强化学习等不同类型：

- 1. 监督学习：**在监督学习中，模型接收带有标签的数据作为输入，目标是学习将输入数据映射到标签的关系。常见的任务包括分类（如图像分类、垃圾邮件检测）和回归（如房价预测、股票价格预测）。
- 2. 无监督学习：**无监督学习是指模型从未标记的数据中学习，目标是发现数据的隐藏结构或模式。常见的任务包括聚类（如客户分群、文档聚类）和降维（如主成分分析）。
- 3. 强化学习：**强化学习是一种通过与环境互动来学习的方法，模型根据其行为的反馈来调整其决策策略。强化学习常用于构建自动控制和决策系统，如自动驾驶汽车和游戏玩家。

机器学习的应用非常广泛，包括自然语言处理、图像识别、推荐系统、金融预测等领域。随着数据量的增加和算法的改进，机器学习在解决各种复杂问题上的效果也在不断提高。

# 提示工程



2024/4/28 18:43:59

如何向六岁的孩子解释什么是机器学习？



2024/4/28 18:44:06

嗨！你知道电脑可以像你一样学习吗？就像你在学校里学习新事物一样，电脑也可以学习。机器学习就是一种让电脑学习的方法，就像魔法一样让电脑变得聪明。当你告诉电脑要做什么，它会用自己学到的知识去做。比如，你给电脑看很多猫的图片，告诉它这是猫，那是猫，它就会学会如何识别猫。所以，机器学习就是电脑学习新东西变得更聪明的魔法！



# 提示工程



解释什么是机器学习。请你详细解释：

1. 机器学习的基本概念；
2. 机器学习的发展历史；
3. 机器学习较其他方法有哪些优点；
4. 提供一个简单的例子，说明机器学习如何被实际应用。



2024/4/28 18:48:13

当然可以帮你解释机器学习！这里是详细的解释：

## 1. 机器学习的基本概念：

机器学习是一种人工智能（AI）的分支，它关注如何通过数据和算法使计算机系统能够从经验中学习并改进性能。换句话说，机器学习使计算机系统可以从数据中学习模式和规律，并且在不需要明确编程的情况下做出预测或做出决策。

## 2. 机器学习的发展历史：

机器学习的发展可以追溯到上世纪五六十年代。最早的机器学习方法主要集中在符号推理和专家系统上。随着时间的推移，统计学习方法如回归和分类开始在机器学习领域占据重要地位。近年来，深度学习技术的发展使得机器学习在图像识别、自然语言处理等领域取得了巨大的进步。

## 3. 机器学习较其他方法有哪些优点：

- 可以处理大规模和复杂的数据，发现其中的模式和规律。
- 能够自动化决策过程，减少人工干预和错误率。
- 具有泛化能力，可以对未知数据进行预测和分类。
- 可以不断学习和改进，适应不断变化的环境和需求。

## 4. 提供一个简单的例子，说明机器学习如何被实际应用：

假设我们要建立一个垃圾邮件过滤器。传统的方法可能是基于规则，比如特定关键词出现的频率或者发件人的信

## 更多学习资源

- **Stanford CS224n (by Chris Manning) :**

- <https://web.stanford.edu/class/cs224n/>

- **Coursera 利用分类和向量空间进行自然语言处理:**

- <https://www.coursera.org/learn/classification-vector-spaces-in-nlp>

谢谢！