- TF-IDF
  - 词、文章、文档
  - TF = 某个词在文章中出现的总次数/文章的总词数
  - IDF = log（词料库的文档总数/包含该词的文档数+1）
  - TF-IDF = TF*IDF

词向量

- 自然语言中的词语在机器学习中表示符号
  - One-hot Representation

word	i	like	enjoy	deep	learning	NLP	flying
i	1	0	0	0	0	0	0
like	0	1	0	0	0	0	0
enjoy	0	0	1	0	0	0	0
deep	0	0	0	1	0	0	0
learning	0	0	0	0	1	0	0
NLP	0	0	0	0	0	1	0
Flying	0	0	0	0	0	0	1

1是词语数量较大时，向量维度高且稀疏，向量矩阵巨大而难以存储。

2是向量并不包含单词的语义内容，只是基于数量统计。

词向量

- Distributional Representation
  - 词表示为：
    - [0.792, −0.177, −0.107, 0.109, 0.542, ...]，常见维度50或者100
  - 解决“词汇鸿沟”问题
    - 可以通过计算向量之间的距离（欧式距离、余弦距离等）来体现词与词的相似性
- 如何训练这样的词向量
  - 没有直接的模型可训练得到
  - 可通过训练语言模型的同时，得到词向量

语言模型

- 判断一句话是不是正常人说出来的，用数学符号描述为
  - 给定一个字符串"w1,w2,...,wt",计算它是自然语言的概率，一个很简单的推论是
  - 例如，有个句子"大家,喜欢,吃,苹果"
    - P(大家，喜欢，吃，苹果)=p(大家)p(喜欢|大家)p(吃|大家,喜欢)p(苹果|大家,喜欢,吃)
  - 简单表示为
- 计算

$$p(w_1, w_2, \ldots, w_t)$$

$$p(w_1, w_2, \ldots, w_t) = p(w_1) \cdot p(w_2 \mid w_1) \cdot p(w_3 \mid w_1, w_2) \cdot \ldots \cdot p(w_t \mid w_1, w_2, \ldots, w_{t-1})$$

$$p(s) = p(w_1, w_2, \ldots, w_T) = \prod_{i=1}^{T} p(w_i \mid Context_i)$$

$$p(w_i \mid Context_i)$$

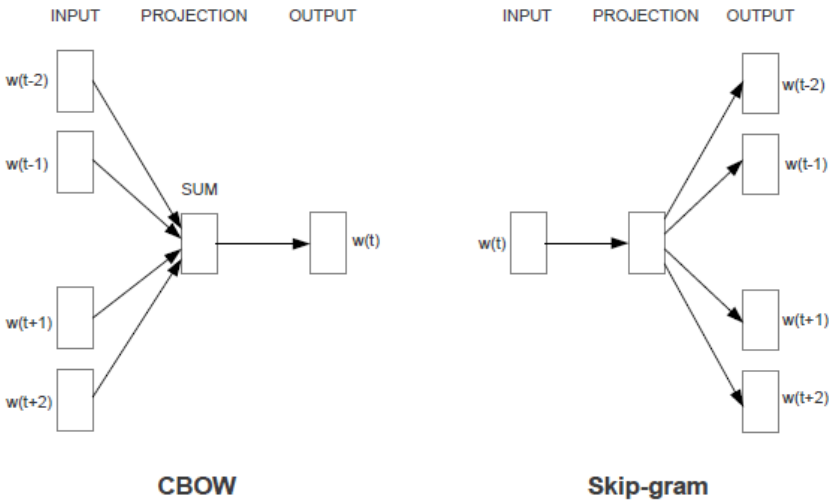
Google的Mikolov在2013年推出了一款计算词向量的工具

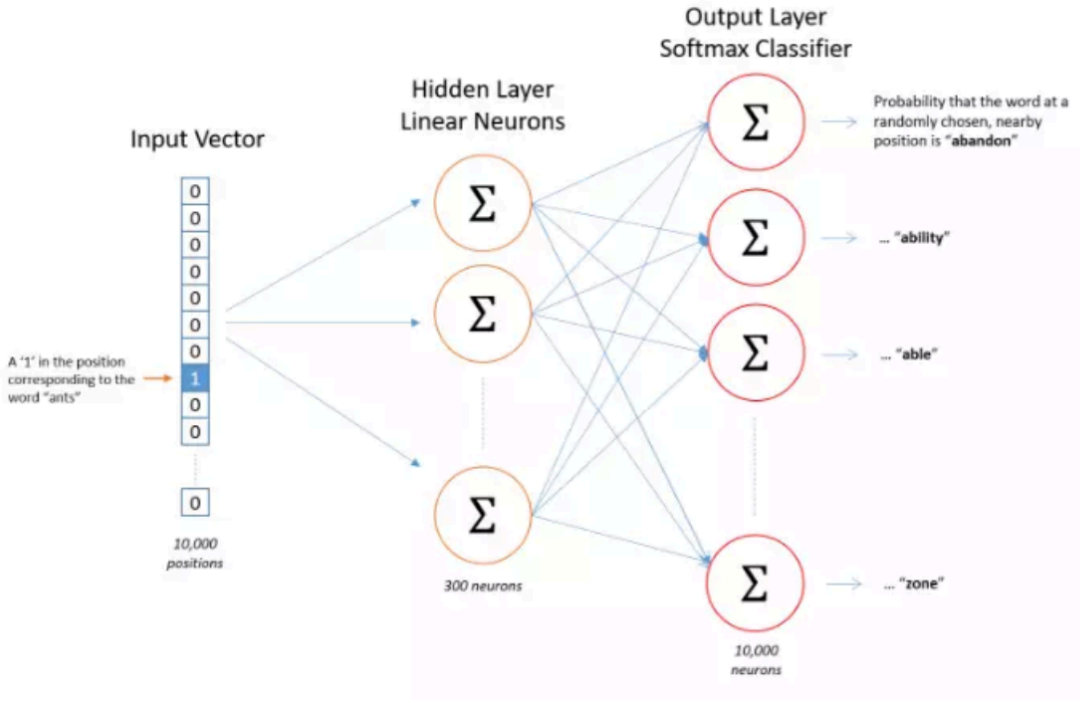
word2vec作为神经概率语言模型的输入，其本身其实是神经概率模型的副产品，是为了通过神经网络学习**某个语言模型**而产生的中间结果。具体来说，“某个语言模型”指的是“CBOW”和“Skip-Gram”。

两个语言模型

CBOW：Continuous Bag-of-Words

Skip-Gram：Continuous Skip-Gram Model

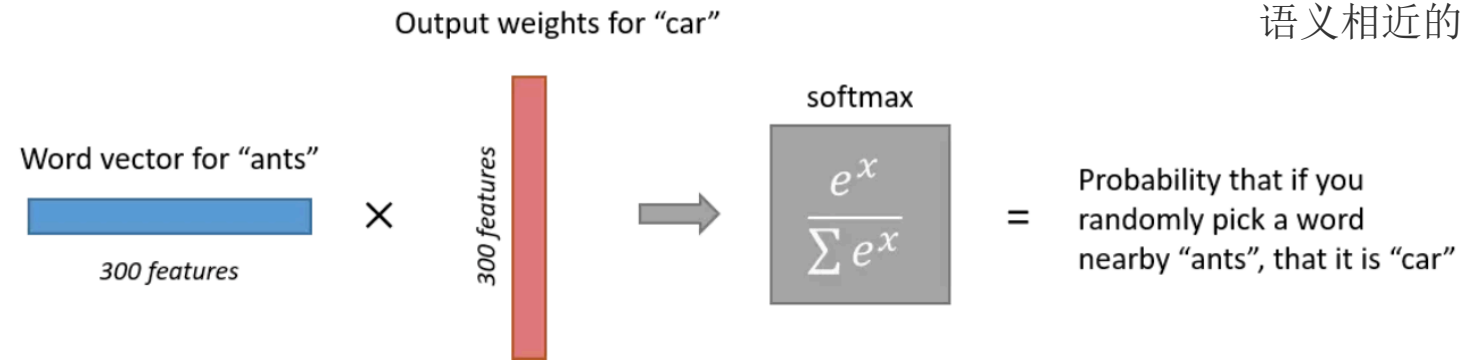




隐藏层：

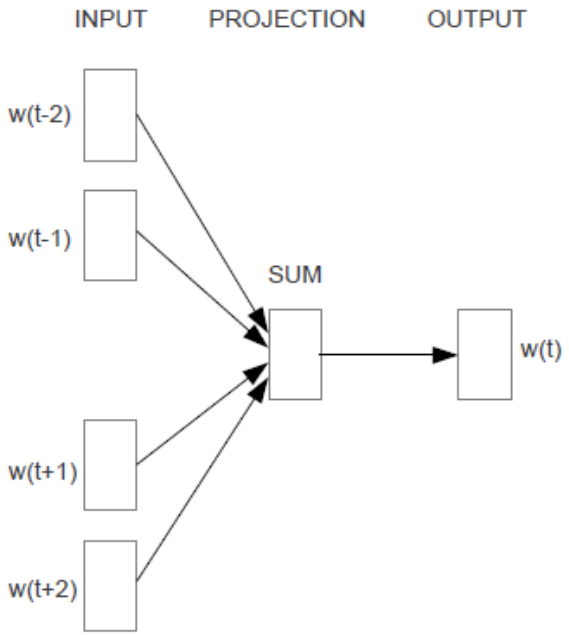
输入　　权重　　隐藏层

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



语义相近的词其词向量也相近。

- CBOW模型
  - INPUT:输入层
  - PROJECTION:投影层
  - OUTPUT:输出层
  - w(t):当前词语（向量）
  - w(t-2),w(t-1),w(t+1),w(t+2):当前词语的上下文
  - SUM:上下文的累加和



CBOW