Word2Vec 提供的两种训练架构与工作原理：
Word2Vec 的两种训练架构：连续词袋模型（CBOW）与跳字模型（Skip-gram）
连续词袋模型（CBOW）：
目标与输入输出：CBOW 的目标是根据上下文词汇来预测中心词汇。输入是某个中心词一定**窗口**内的上下文词汇的词向量，输出是中心词汇的词向量。从概率模型角度，目标是最大化$P(w_t|w_{t-n},w_{t-n+1},\dots,w_{t-1},w_{t+1},w_{t+2},\dots,w_{t+n})$
流程：首先对输入的上下文词汇的 one-hot 向量（初始时采用，后续会用训练好的词向量迭代更新）进行**嵌入矩阵**查找，得到对应的**词向量**，然后将这些上下文词向量进行平均，得到一个**上下文语义表示向量**。接着通过一个**线性变换**和一个 **softmax 激活函数**，得到对中心词的**概率分布预测**。最后通过对比真实中心词汇的词向量，利用损失函数，计算预测误差，并通过反向传播算法更新嵌入矩阵和输出权重矩阵中的参数，从而使得预测更为精确。
跳字模型（Skip-gram）：
目标与输入输出：与 CBOW 相反，Skip-gram 是根据中心词汇来预测上下文词汇。输入是中心词汇的词向量，输出是其周围一定窗口内上下文词汇的词向量。对应的概率模型是最大化$P(w_{t-n},w_{t-n+1},\dots,w_{t-1},w_{t+1},\dots,w_{t+n}|w_t)$
流程：首先对输入的中心词汇的 one-hot 向量进行**嵌入矩阵**查找，得到对应的**词向量**。然后将该中心词汇通过**线性变换和 softmax 激活函数**，为每个可能的上下文词汇计算**预测概率**，随后同样将真实的上下文词汇的 one-hot 向量作为标签，通过损失函数**计算误差并且反向传播**，来更新嵌入矩阵等参数。训练完成后，嵌入矩阵中的向量即为**词向量**。

利用 Gensim 的训练结果：

```
与‘computer’最相似的5个词
treaty: 0.9548
intake: 0.9454
excellence: 0.9443
tube: 0.9440
facility: 0.9435
'apple' 和 'fruit' 的相似度： 0.8328
'king' - 'man' + 'woman' 最接近的词： sold（相似度： 0.9527）
```