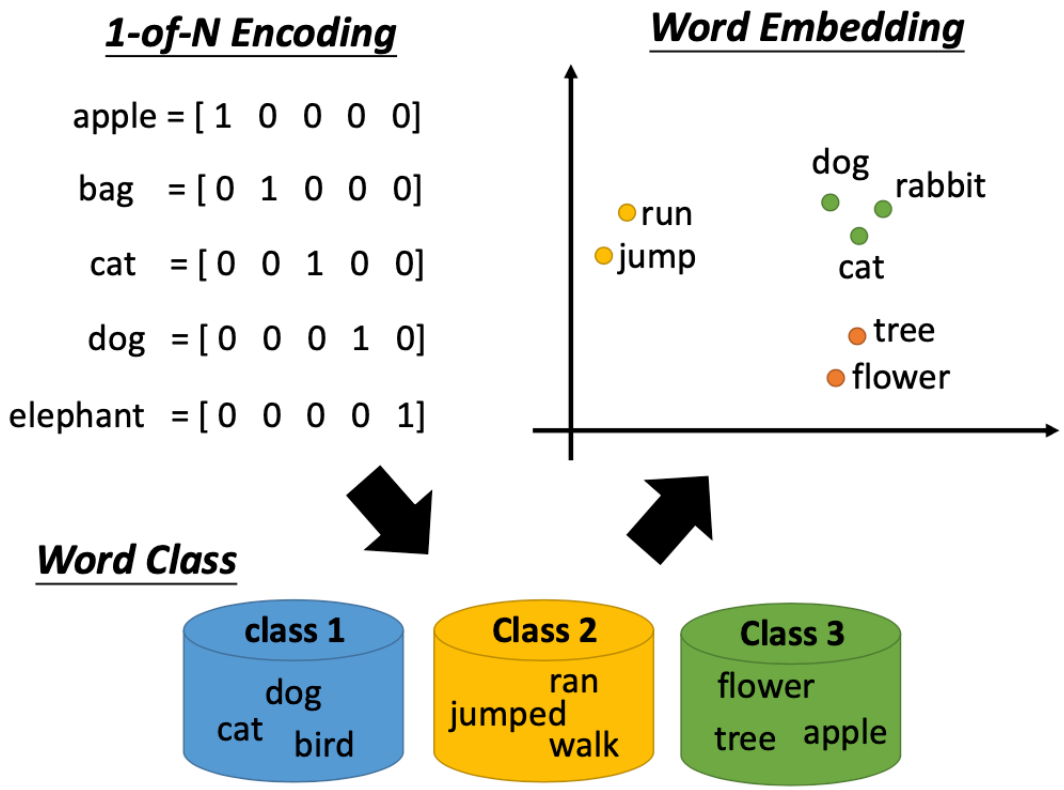
Word Embedding

如果使用1-of-N encoding，每个单词用一个vector表示，那么5个单词就需要5个vector；如果我们把同一个类别的单词都放到那个类里，即属于class1的单词有dog、cat、bird，属于动物类的单词，同理可以得出class2，class3；

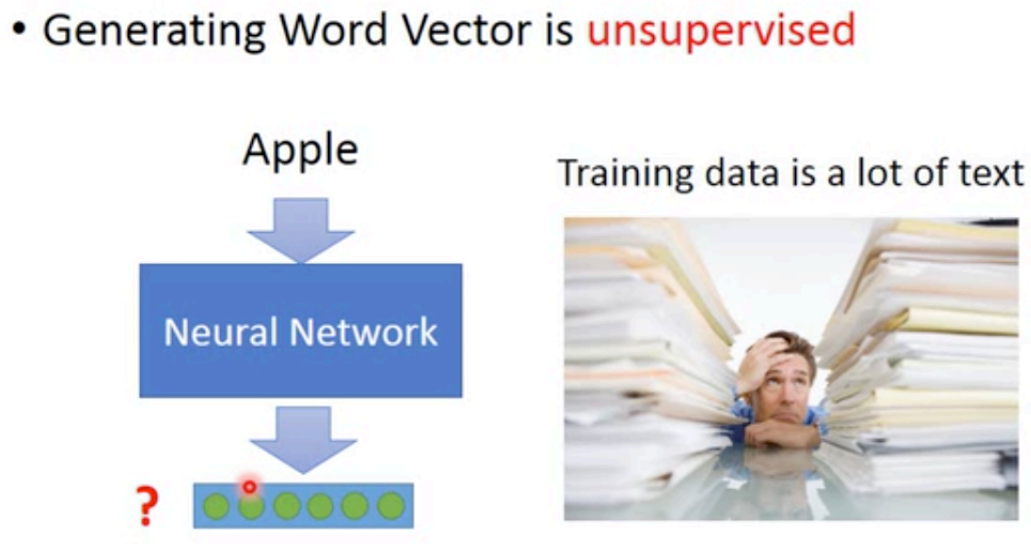
但只做classify是不够的，这些class之间也有一些其他的联系；比如class1属于动物，class3属于植物，他们都是生物，只做classify并不能体现这种联系；

现在我们把每个word都project到一个两个dimension上， 水平的dimension可以是表示生物（class1，class3）和其他类别class2之间的差距，竖直的dimension可以是会动（class2，class3）的和不会动class1之间的差距；

如果现在有10w个单词，1-of-N encoding就需要10w个vector，但word embedding可能只需要50维左右，就可以表示这些所有的word。



word2vec是一个无监督学习问题，如果network的input为“Apple”，要输出其对应的vector

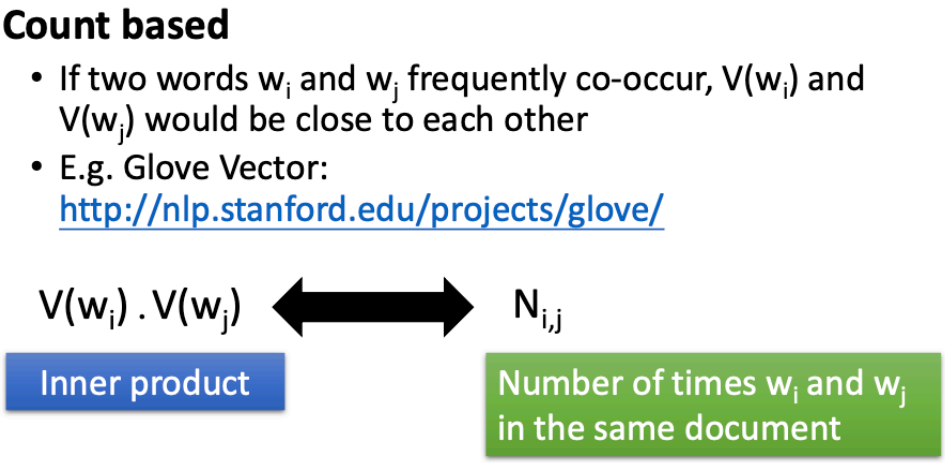


下面我们将叙述生成词向量的两种主要手段。

Count based

基于计数的方法，记录文本中词的出现次数。如果两个单词w_i, w_j常常一起出现，那么我们就认为其对应的vector $V(w_i), V(w_j)$之间就是非常接近的。

用$N_{i,j}$表示w_i, w_j在同一个document中出现的次数，那么我们希望找到对应的$V(w_i), V(w_j)$，其做inner product的值和这个次数越接近越好。

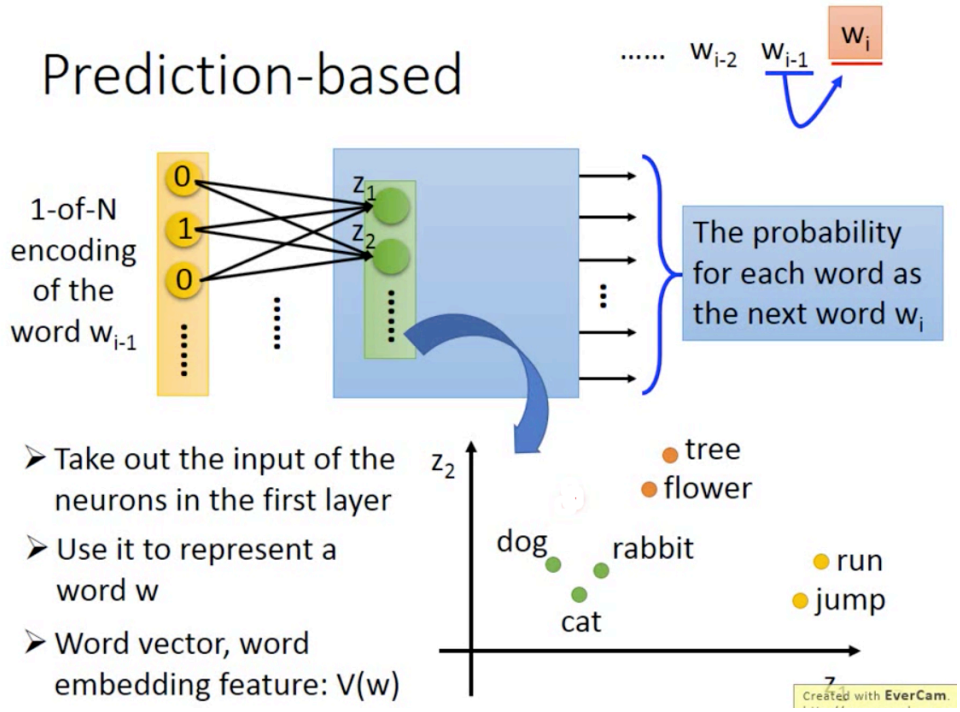


Prediction-based

基于预测的方法，即可以通过上下文预测中心词，也可以通过中心词预测上下文。中心词即我们要预测的词。在下文中，w_i是我们要预测的值，$\ldots, w_{i-2}, w_{i-1}$就是其对应的上下文；

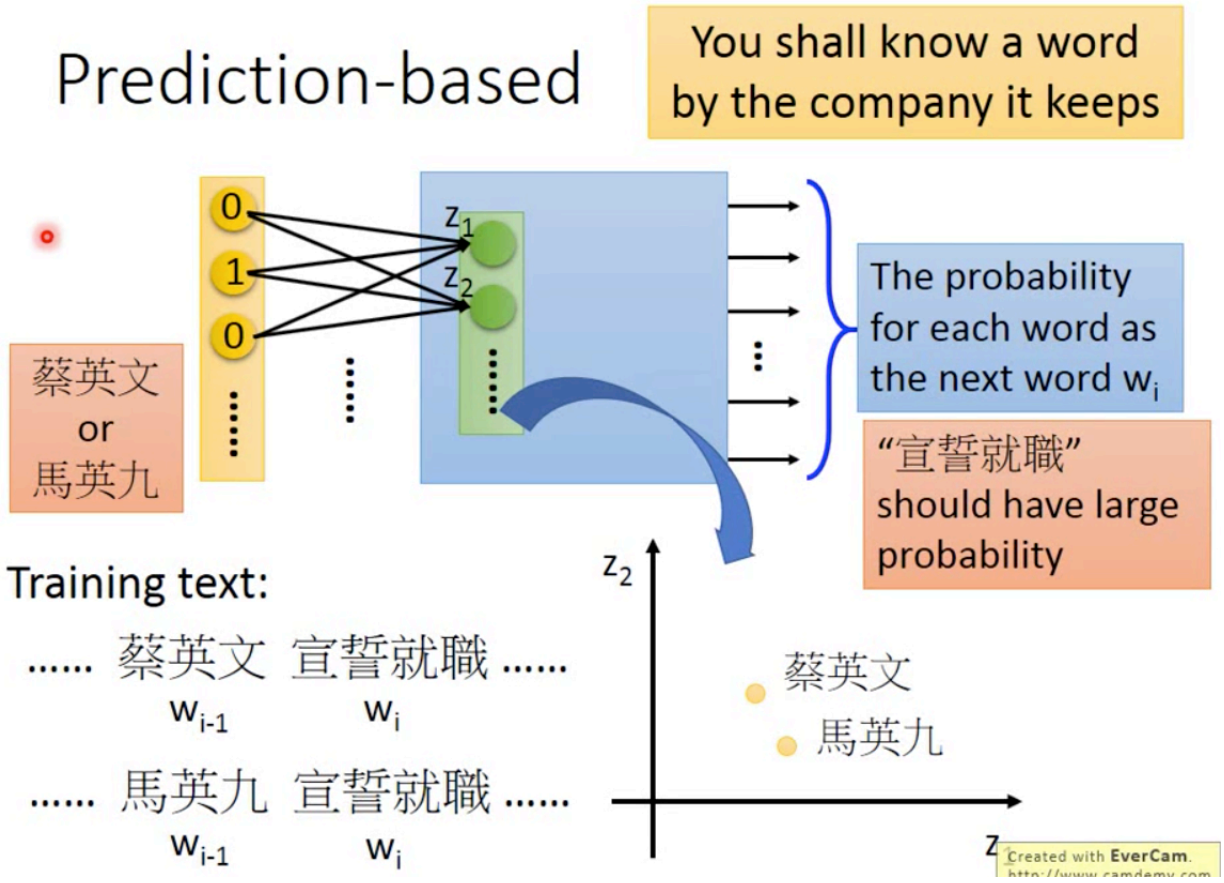
现在我们把w_{i-1}的one-hot encoding作为网络的输入，网络的输出为每个词作为下一个词w_i输出的概率，如果词袋中有10w个词，那么输出的维度就对应为10w维。

把网络中第一个hidden layer的input拿出来，即$z = (z_1, z_2, \ldots)^T$；如果输入为不同1-of-N encoding，那么对应的z也是不一样的。我们就可以用z来代表一个word，即z就是我们要寻找的word vector $V(w)$。



如果现在有两个training text，“蔡英文 宣誓就职”、“马英九 宣誓就职”，如果输入的1-of-N encoding是“蔡英文”或“马英九”，那么我们希望在网络的输出概率中，“宣誓就职”的概率是最大的。同理我们也可以把网络的第一个hidden layer的输入作为z，就是我们要寻找的word vector $V(w)$。

这时我们就需要中间的hidden layer来做这样一件事，如果输入为不同的词汇（“蔡英文”，“马英九”），那么我们希望中间的hidden layer可以把不同的词汇project到相同的空间，这样网络的输出才可能都是“宣誓就职”对应的概率最大。



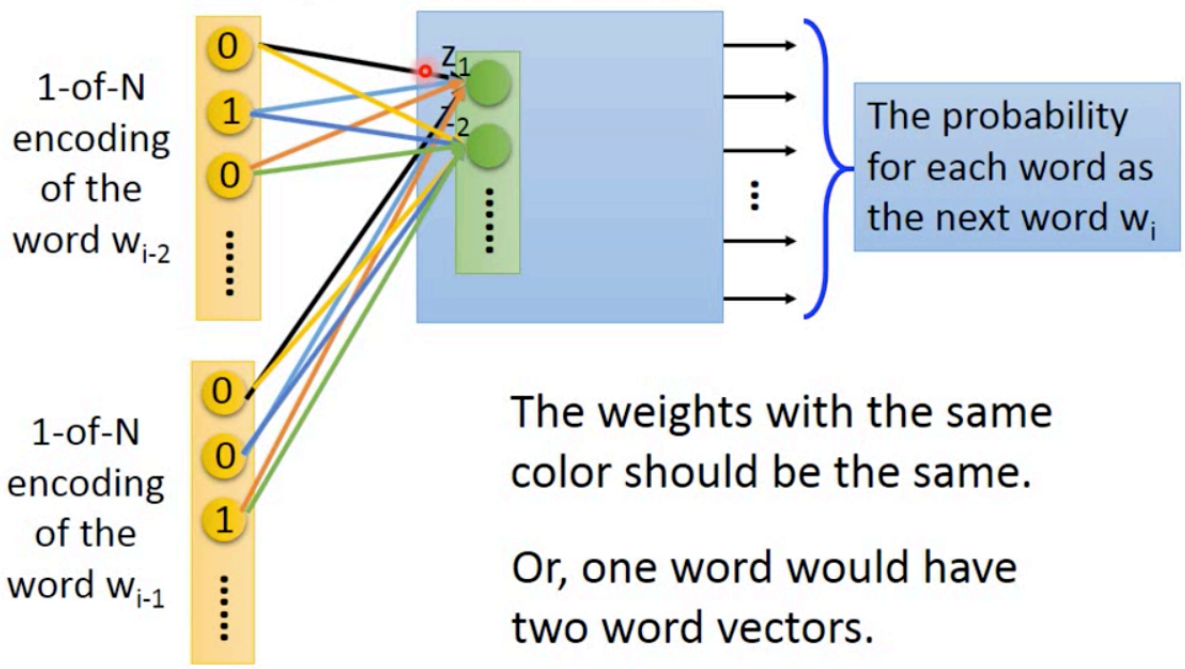
Sharing weight

只考虑前面的一个词汇，来预测下一个词汇会很难，我们可以考虑前面的几个词汇。现在我们来叙述考虑前两个词汇的结果。

现在我们并不能像之间的neural network那样，所有的输入都连成一个vector作为输入。但实际上，我们希望不同的one-hot vector的同一维度之间是tie在一起的。即w_{i-1}, w_{i-2}的第一维对应z_1，其对应的weight是一样的；同理w_{i-1}, w_{i-2}的第二维对应z_1，其对应的weight是一样的，.....

Q：为什么不同的one-hot vector的同一维度之间是tie在一起的呢?

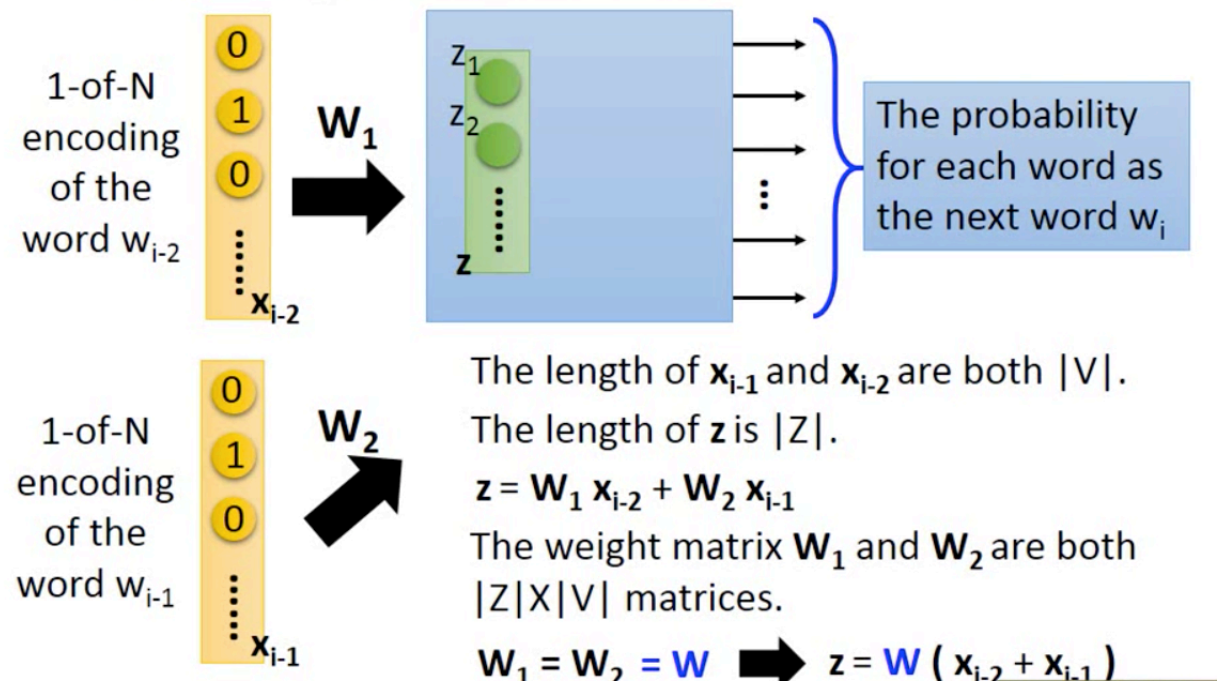
A：我们现在把同一个word放到w_{i-1}, w_{i-2}的位置， 如果每一维对应的weight都不一样，那么实际上就输入了两个vector。



如果我们设置x_{i-1}, x_{i-2}的长度都是$|V|$，z的长度是$|Z|$，那么

$$z = W_1 x_{i-2} + W_2 x_{i-1}$$

其中$W_1 = W_2 = W$，那么$z = W(x_{i-2} + x_{i-1})$，也就得到了word vector $V(w)$，



那么在实际的网络训练中，我们如何保证$W_1 = W_2$呢?

首先需要将w_1, w_2初始化为相同的值，那么

$$w_i \leftarrow w_i - \eta \frac{\partial C}{\partial w_i}$$
$$w_j \leftarrow w_j - \eta \frac{\partial C}{\partial w_j}$$

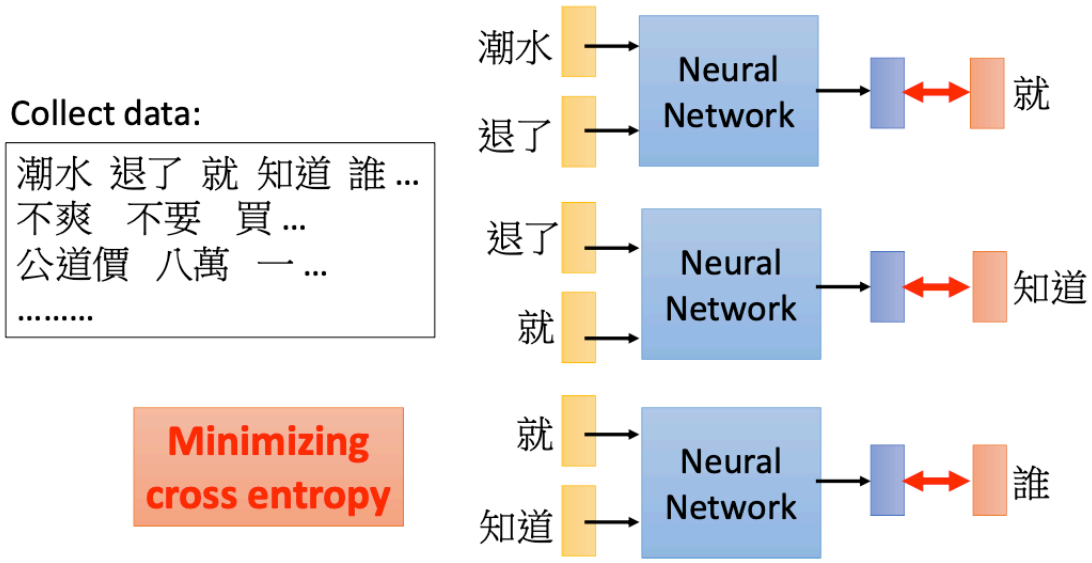
由于gradient的值不同，w_1, w_2在更新一次参数之后就不相等了，必须保证每次更新之后的值还是一样的，因此

$$w_i \leftarrow w_i - \eta \frac{\partial C}{\partial w_i} - \eta \frac{\partial C}{\partial w_j}$$
$$w_j \leftarrow w_j - \eta \frac{\partial C}{\partial w_j} - \eta \frac{\partial C}{\partial w_i}$$

这样每次更新的值都一样的，也就保证了$w_1 = w_2$

Training

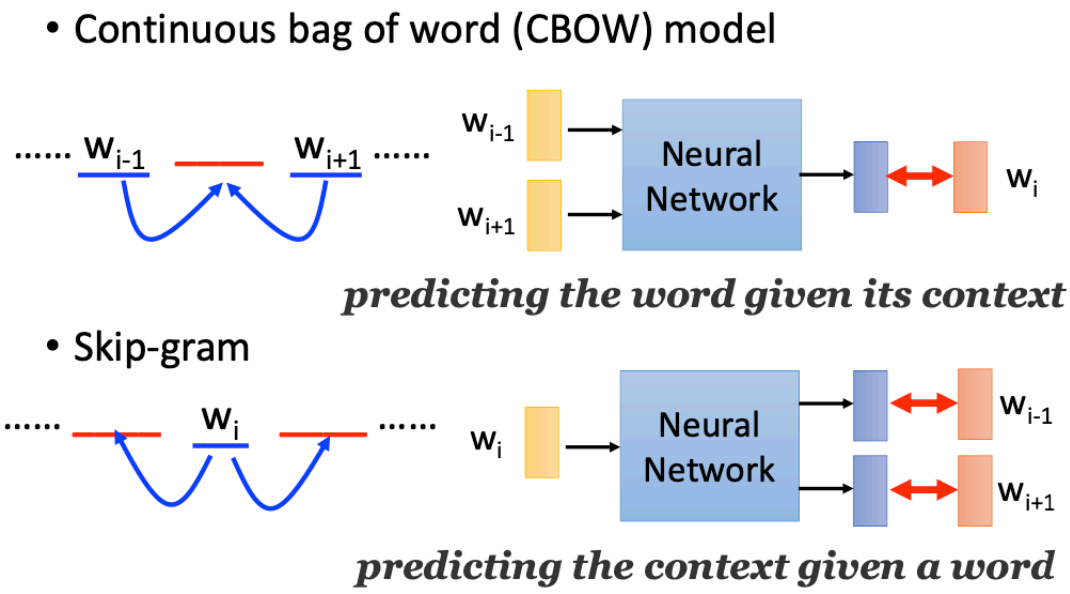
对于输入“潮水、退了”，我们希望network的输出和“就”越接近越好，即最小化cross entropy



Various Architectures

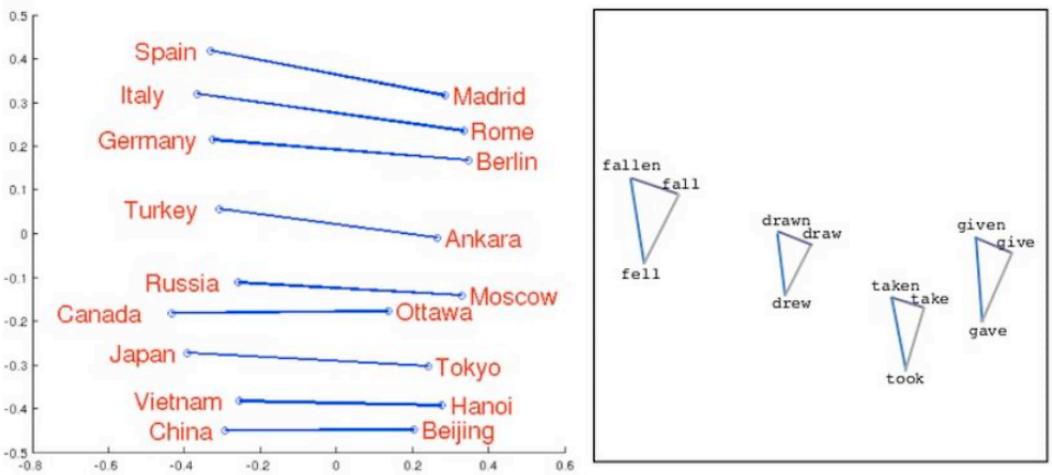
CBOW：根据上下文的词汇w_{i-1}, w_{i+1}来预测中心词w_i;

Skip-gram：根据中心词w_i来预测上下文w_{i-1}, w_{i+1}。



Result

vec(Rome) - vec(Italy) ≈ vec(Berlin) - vec(Germany)，Italy和Rome之间有is-capital-of的关系，这种关系也恰好在Madrid和Spain之间出现。



如果现在有人问机器一个问题，Rome和Italy之间的关系就像是Berlin和什么的关系？我们就可以通过计算vec(Berlin) ≈ vec(Rome) - vec(Italy) + vec(Germany)得出结果。

- Characteristics

$$V(Germany) \approx V(Berlin) - V(Rome) + V(Italy)$$

$$V(hotter) - V(hot) \approx V(bigger) - V(big)$$
$$V(Rome) - V(Italy) \approx V(Berlin) - V(Germany)$$
$$V(king) - V(queen) \approx V(uncle) - V(aunt)$$

- Solving analogies

Rome : Italy = Berlin : ?

Compute $V(Berlin) - V(Rome) + V(Italy)$

Find the word w with the closest V(w)