01 Introduction and Word Vectors

在这里，我们介绍 Word2Vec 算法。

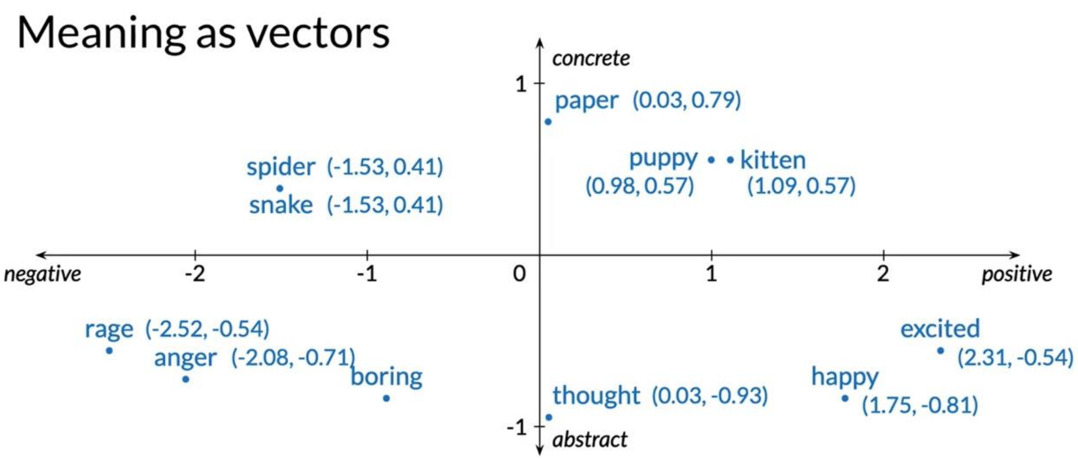
Word2Vec 由 Google 于 2013 年提出，是一个学习单词向量的框架。在传统的自然语言处理中，我们往往把单词看作是离散的符号，单词可以用 one-hot 编码表示，例如：

$$motel = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$$
$$hotel = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

它的缺点是非常明显的：一、所有的向量都是正交的，无法通过向量表示出单词之间的相关性；二、每个单词的维度过大，容易出现内存不足或稀疏性等现象。

在 Word2Vec 算法中，单词向量的维度由人为控制，并且可以通过单词向量来分析单词间的相似性和关系。

举一个简单的例子，如图一所示。如果把每个单词以二维向量表示，其中x轴代表单词的 positive、negative 程度，y轴表示单词的 concrete、abstract 程度。我们可以发现，spider 和 snake 是两个相近的单词。



图一

对于 Word2Vec 算法，在这里定义：

c：一个中心词

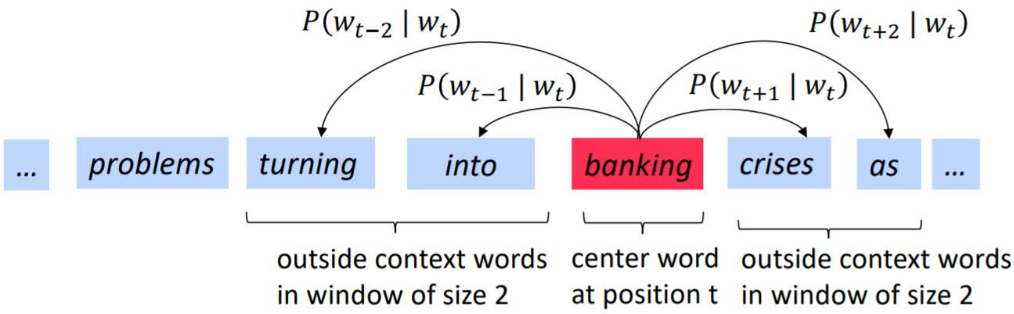
o：c的一个上下文词

对于每个单词 w，我们制定两个向量，其中：

当 w 为中心词时，向量为v_w

当 w 为上下文词时，向量为u_w

在图二中，定义了窗口大小和中心词的概念。在接下来的分析中，我们会直接使用这两个概念。



图二

Word2Vec 算法执行的目的，是最大化给定c后的o的概率（反之亦然），我们通过梯度下降法（gradient descent）不断调整词向量，以最大化这个概率。

对于每个位置$t = 1, \dots, T$，在大小为m的固定窗口内预测上下文单词。给定中心词w_j。

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

其中θ为所有需要被优化的变量。

定义损失函数

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

现在，我们需要定义计算$P(w_{t+j}|w_t)$的公式，我们定义

$$P(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w \in V} \exp\left(u_w^T v_c\right)}$$

$u^T v$是词向量点乘，向量越相似，乘积也就越大。同时，$P(o|c)$是一个$softmax$函数，可以起到两个作用：1、放大最大的概率；2、为较小的值赋予概率，如为0（两个向量正交）赋予概率。

在随机初始化u_w和v_w后，用梯度下降法（gradient descent）进行更新。

1、对u_o进行更新，如图三所示。

在持续更新后，理论上会使

$$\frac{\partial \log P(o|c)}{\partial u_o} \approx 0$$

因此$P(o|c) \approx 1$，即通过中心词c，我们可以正确预测上下文词o。

$$
\begin{aligned}
\frac{\partial}{\partial u_o}\log P(o|c) &= \frac{\partial}{\partial u_o}\log\frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)} \\
&= \frac{\partial}{\partial u_o}\left(\log\exp(u_o^T v_c) - \log\sum_{w\in V}\exp(u_w^T v_c)\right) \\
&= \frac{\partial}{\partial u_o}\left(u_o^T v_c - \log\sum_{w\in V}\exp(u_w^T v_c)\right) \\
&= v_c - \frac{\sum \frac{\partial}{\partial u_o}\exp(u_w^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)} \\
&= v_c - \frac{\exp(u_o^T v_c)v_c}{\sum_{w\in V}\exp(u_w^T v_c)} \\
&= v_c - \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}v_c \\
&= v_c - P(o|c)v_c \\
&= (1 - P(o|c))v_c
\end{aligned}
$$

图三

2、对v_c进行更新，如图四所示。

$$
\begin{aligned}
\frac{\partial}{\partial v_c}\log P(o|c) &= \frac{\partial}{\partial v_c}\log\frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)} \\
&= \frac{\partial}{\partial v_c}\left(\log\exp(u_o^T v_c) - \log\sum_{w\in V}\exp(u_w^T v_c)\right) \\
&= \frac{\partial}{\partial v_c}\left(u_o^T v_c - \log\sum_{w\in V}\exp(u_w^T v_c)\right) \\
&= u_o - \frac{\sum_{w\in V}\exp(u_w^T v_c)u_w}{\sum_{w\in V}\exp(u_w^T v_c)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial v_c}\log P(o|c) &= u_o - \frac{\sum_{w\in V}\exp(u_w^T v_c)u_w}{\sum_{w\in V}\exp(u_w^T v_c)} \\
&= u_o - \sum_{w\in V}\frac{\exp(u_w^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}u_w \\
&= u_o - \sum_{w\in V}P(w|c)u_w
\end{aligned}
$$

图四

同理，在持续更新后，理论上会使

$$u_o - \sum_{w\in V}P(w|c)u_w \approx 0$$

即

$$u_o \approx \sum_{w \in V} P(w|c) u_w$$

而

$$\sum_{w \in V} P(w|c) u_w$$

可看作给定中心词c后，所预测的上下文词的词向量u_w的加权平均。
因此可知在经过梯度下降法后，模型预测的上下文词将逐步接近真正的上下文词。

参考文献

[1] https://looperxx.github.io/CS224n-2019-01-Introduction%20and%20Word%20Vectors/

[2] https://www.coursera.org/learn/probabilistic-models-in-nlp