Tensorflow-覃秉丰

Word2vec：

当我们分析图片或者语音的时候，我们通常都是在分析密集的，高纬度的数据集。我们所需的全部信息都储存在原始数据中。

```
[[ 0.65337904  0.96147407  0.89736144  0.97613636  0.53563182  0.65046753]
 [ 0.22471787  0.67623082  0.29457548  0.54820279  0.25811241  0.10811792]
 [ 0.15491558  0.4922566   0.94136616  0.18930393  0.43129747  0.0312585 ]
 [ 0.32249593  0.13105882  0.55929974  0.60043924  0.09488365  0.93599279]
 [ 0.18468721  0.80349133  0.77069437  0.34970681  0.04205231  0.07288426]
 [ 0.9713573   0.31079413  0.60528272  0.24704021  0.82908679  0.78950803]
 [ 0.92664684  0.77715744  0.55786552  0.85356888  0.19111345  0.20953576]
 [ 0.02344845  0.57778919  0.65908075  0.4059088   0.0907254   0.06996104]
 [ 0.72560051  0.91087261  0.66252184  0.06852047  0.56545598  0.40305866]
 [ 0.80040794  0.60398618  0.07660456  0.22238826  0.65349584  0.53116871]
 [ 0.41366496  0.30961498  0.78078967  0.21373827  0.11872793  0.13299166]
 [ 0.73777544  0.13902513  0.48004225  0.683896    0.20811546  0.30064903]
 [ 0.76508436  0.85263635  0.16590127  0.18754474  0.86105624  0.41046465]
 [ 0.37545851  0.02911257  0.27524078  0.00883495  0.53383195  0.72747815]
 [ 0.27355726  0.85399793  0.70522708  0.86964774  0.31380896  0.14360617]
 [ 0.92621366  0.81976771  0.34924696  0.11268561  0.15834104  0.25493069]
 [ 0.86482453  0.66849716  0.81176577  0.73482012  0.72419957  0.61101592]
 [ 0.65271702  0.22533039  0.25093796  0.90895525  0.56729463  0.15508486]
 [ 0.3398385   0.43496739  0.0772549   0.38408786  0.06412806  0.8306255 ]
 [ 0.63361307  0.20169828  0.36050179  0.38680661  0.63106815  0.03255401]]
```

当我们处理自然语言问题的时候，我们通常会做分词，然后给每一个词一个编号，比如猫的编号是120，狗的编号是343。比如女生的编号是1232，女王的编号是2329。这些编号是没有规律，没有联系的，我们从编号中不能得到词与词之间的相关性。

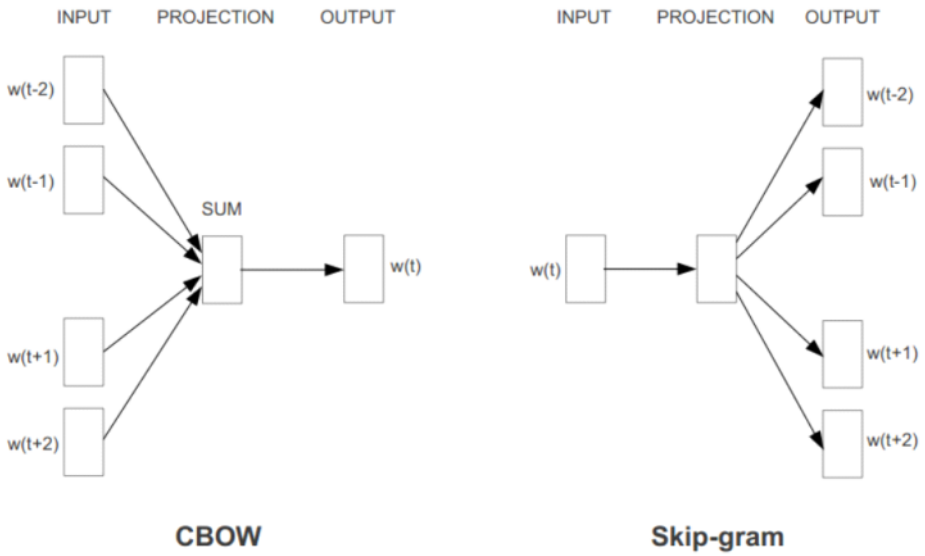
例如：How are you ?
How : 234
Are : 7
you : 987
000...1000000...
00000001000...
000...0000010

CBOW和skip-Gram：

连续词袋模型（CBOW）：
根据词的上下文词汇来预测目标词汇，例如上下文词汇是“今天早餐吃__”，要预测的目标词汇可能是“面包”。

Skip-Gram模型：

Skip-Gram模型刚好和CBOW相反，它是通过目标词汇来预测上下文词汇。例如目标词汇是“早餐”，上下文词汇可能是“今天”和“吃面包”。

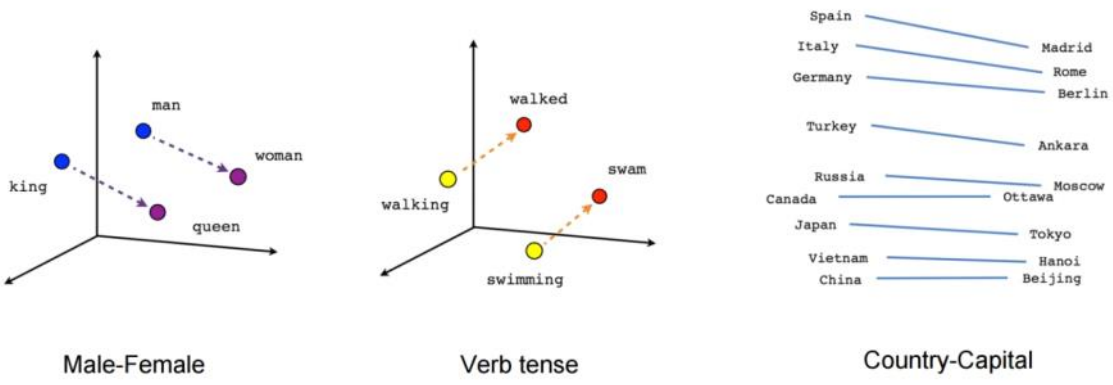


对于这两种模型的训练，我们可能容易想到，使用softmax作为输出层来训练网络。这个方法是可行的，只不过使用softmax作为输出层计算量将会是巨大的。假如我们已知上下文，需要训练模型预测目标词汇，假设总共有50000个词汇，那么每一次训练都需要计算输出层的50000个概率值。

所以训练Word2vec模型我们通常可以选择使用噪声对比估计（Noise Contrastive Estimation）。 NCE使用的方法是把上下文h对应地正确的目标词汇标记为正样本（D=1），然后再抽取一些错误的词汇作为负样本（D=0）。然后最大化目标函数的值。

$$J_{\text{NEG}} = \log Q_\theta(D=1|w_t,h) + k \underset{\tilde{w}\sim P_{\text{noise}}}{\mathbb{E}} [\log Q_\theta(D=0|\tilde{w},h)]$$

当真实的目标单词被分配到较高的概率，同时噪声单词的概率很低时，目标函数也就达到最大值了。计算这个函数时，只需要计算挑选出来的k个噪声单词，而不是整个语料库。所以训练速度会很快。

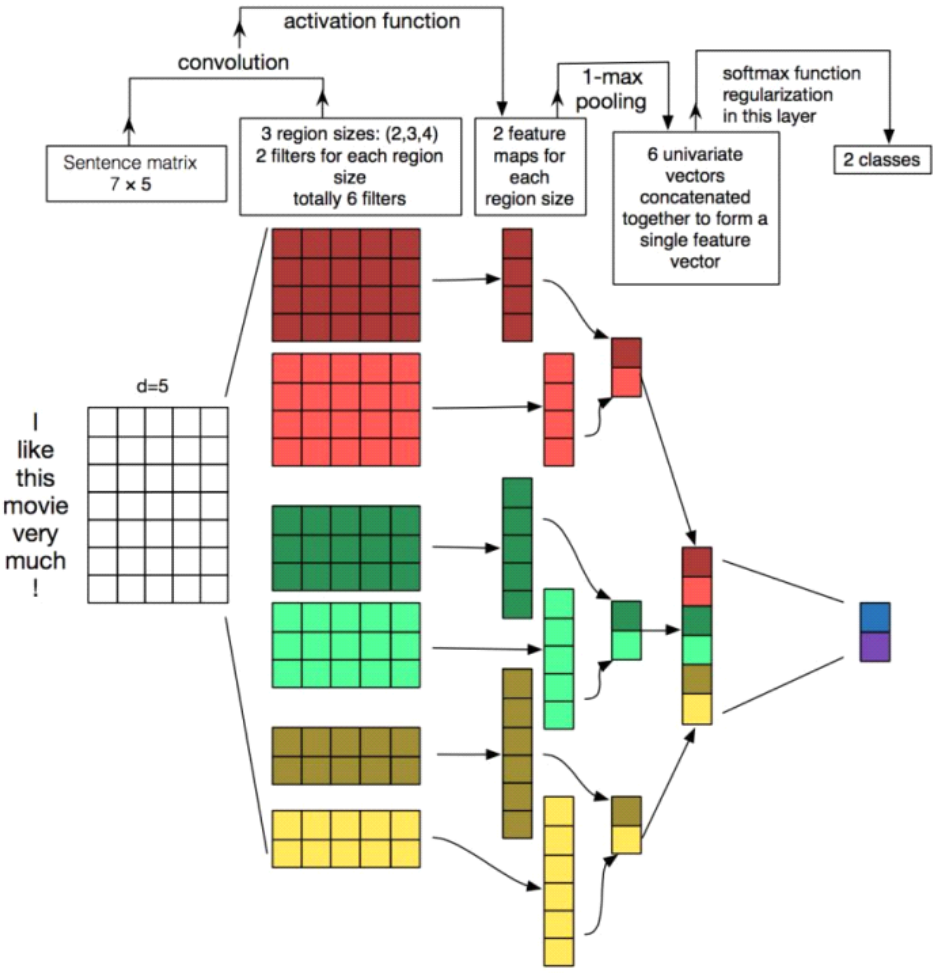


Male-Female　　Verb tense　　Country-Capital

CNN在自然语言处理中的应用：

说到CNN我们首先可能会想到CNN在计算机视觉中的应用。近几年CNN也开始应用于自然语言处理，并取得了一些引人注目的成绩。

CNN应用于NLP的任务，处理的往往是以矩阵形式表达的句子或文本。矩阵中的每一行对应于一个分词元素，一般是一个单词，也可以是一个字符。也就是说每一行都是一个词或者字符的向量（比如前面说到的word2vec）。假设我们一共有10个词，每个词都用128维的向量来表示，那么我们就可以得到一个10×128维的矩阵。这个矩阵就相当于是一副“图像”。



开源项目：

https://github.com/dennybritz/cnn-text-classification-tf

微信公众号：深度学习与神经网络

Github：https://github.com/Qinbf

优酷视频：http://i.youku.com/sdxxqbf