原创 keras：3)Embedding层详解

2017年08月26日 08:54:04 PJ-Javis 阅读数 56680 更多

Embedding层

```
keras.layers.embeddings.Embedding(input_dim, output_dim, embeddings_initializer='uniform',
embeddings_regularizer=None, activity_regularizer=None, embeddings_constraint=None, mask_zero=False,
input_length=None)
```

嵌入层将正整数（下标）转换为具有固定大小的向量，如[[4],[20]]->[[0.25,0.1],[0.6,-0.2]]

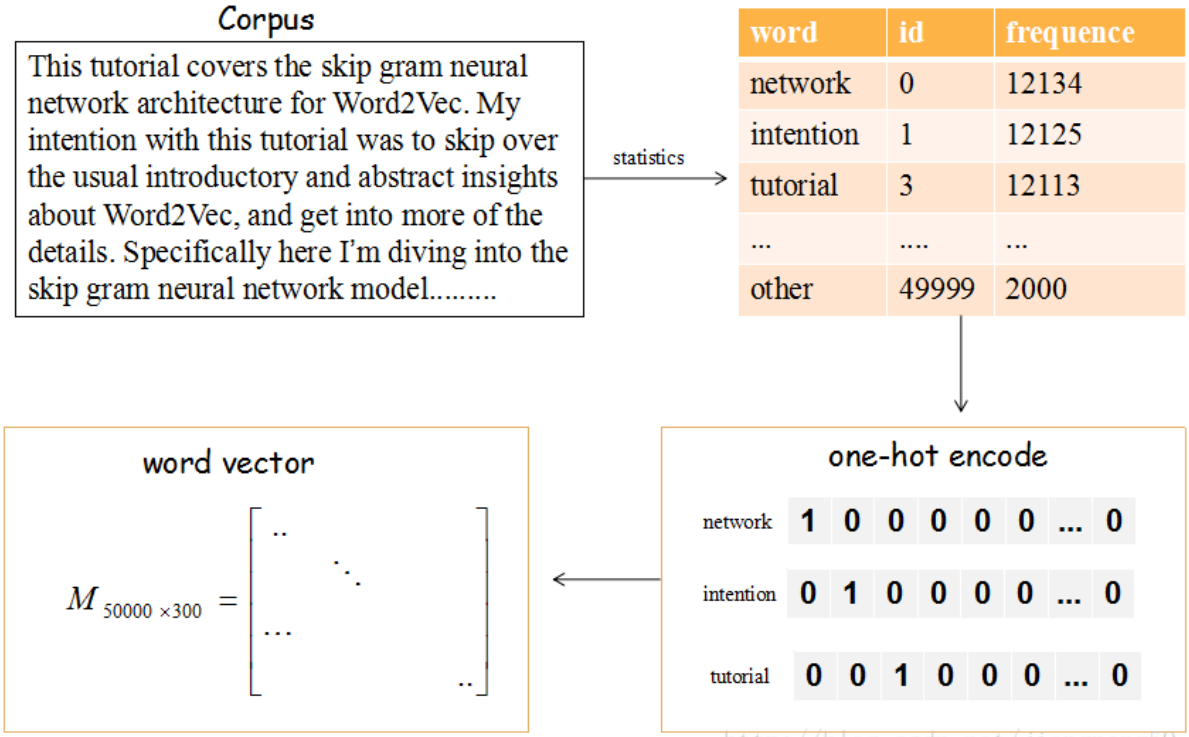
Embedding层只能作为模型的第一层

参数

input_dim：大或等于0的整数，字典长度，即输入数据最大下标+1
output_dim：大于0的整数，代表全连接嵌入的维度
embeddings_initializer: 嵌入矩阵的初始化方法，为预定义初始化方法名的字符串，或用于初始化权重的初始化器。参考initializers
embeddings_regularizer: 嵌入矩阵的正则项，为Regularizer对象
embeddings_constraint: 嵌入矩阵的约束项，为Constraints对象
mask_zero：布尔值，确定是否将输入中的‘0’看作是应该被忽略的‘填充’（padding）值，该参数在使用递归层处理变长输入时有用。设置为True的话，模型中后续的层必须都支持masking，否则会抛出异常。如果该值为True，则下标0在字典中不可用，input_dim应设置为|vocabulary| + 2。
input_length：当输入序列的长度固定时，该值为其长度。如果要在该层后接Flatten层，然后接Dense层，则必须指定该参数，否则Dense层的输出维度无法自动推断。
输入shape
形如（samples，sequence_length）的2D张量
输出shape
形如(samples, sequence_length, output_dim)的3D张量

较为费劲的就是第一句话：
嵌入层将正整数（下标）转换为具有固定大小的向量，如[[4],[20]]->[[0.25,0.1],[0.6,-0.2]]

哪到底咋转啊，亲?
这涉及到词向量，具体看可以参考这篇文章：Word2vec 之 Skip-Gram 模型，下面只进行简单的描述，



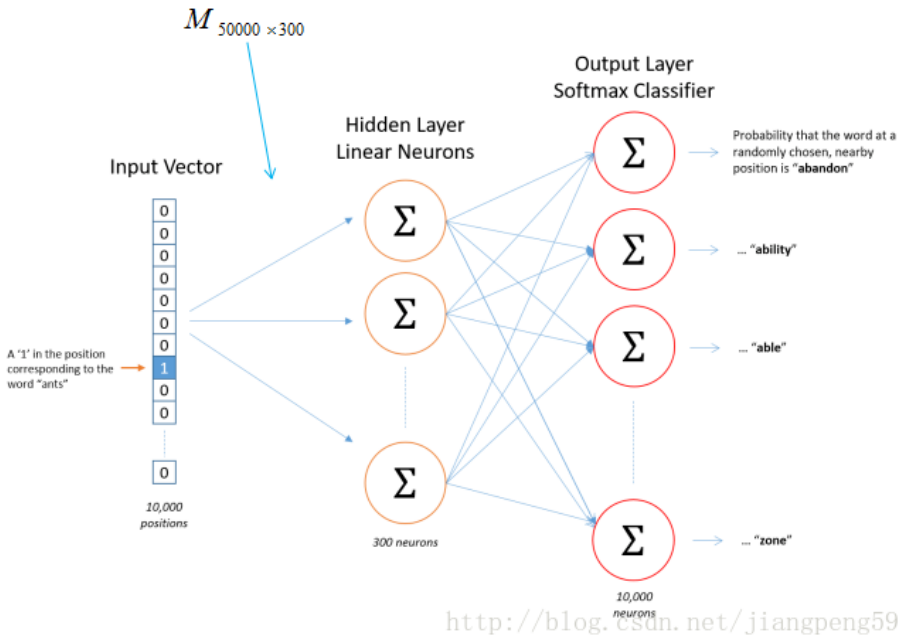
上图的流程是把文章的单词使用词向量来表示。

(1)提取文章所有的单词，把其按其出现的次数降许(这里只取前50000个)，比如单词‘network’出现的次数最多，编号ID为0，依次类推...

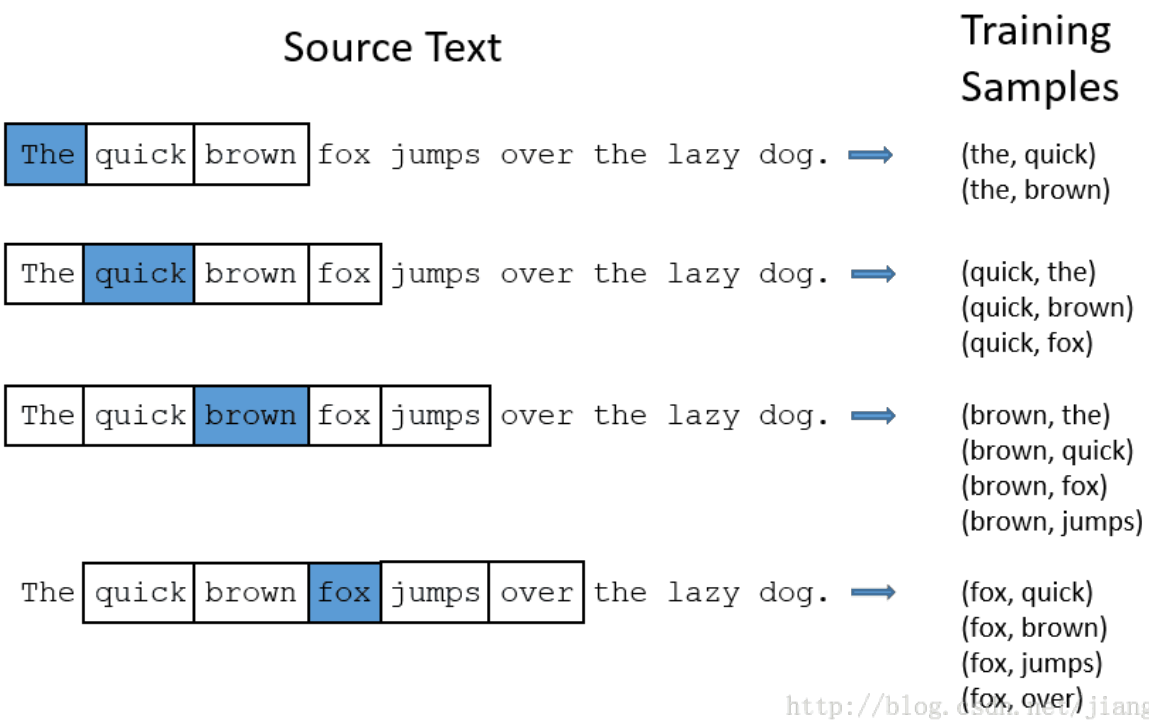
(2)每个编号ID都可以使用50000维的二进制(one-hot)表示

(3)最后，我们会生产一个矩阵M，行大小为词的个数50000，列大小为词向量的维度(通常取128或300)，比如矩阵的第一行就是编号ID=0，即network对应的词向量。

那这个矩阵M怎么获得呢？在Skip-Gram 模型中，我们会随机初始化它，然后使用神经网络来训练这个权重矩阵



那我们的输入数据和标签是什么？如下图，输入数据就是中间的哪个蓝色的词对应的one-hot编码，标签就是它附近词的one-hot编码(这里windown_size=2,左右各取2个)



就上述的Word2Vec中的demo而言，它的单词表大小为1000，词向量的维度为300，所以Embedding的参数input_dim=10000，output_dim=300

回到最初的问题：***嵌入层将正整数（下标）转换为具有固定大小的向量，如[[4],[20]]->[[0.25,0.1],[0.6,-0.2]]***

举个栗子：假如单词表的大小为1000，词向量维度为2，经单词频数统计后，tom对应的id=4，而jerry对应的id=20，经上述的转换后，

我们会得到一个$M_{1000\times2}$的矩阵，而tom对应的是该矩阵的第4行，取出该行的数据就是[0.25,0.1]

如果输入数据不需要词的语义特征语义，简单使用Embedding层就可以得到一个对应的词向量矩阵，但如果需要语义特征，我们大可把训练好的词向量权重直接扔到Embedding层中即可，具体看参考keras提供的栗子:在Keras模型中使用预训练的词向量