## 1.word embedding

[https://www.tensorflow.org/tutorials/word2vec](https://www.tensorflow.org/tutorials/word2vec%E7%9A%84%E5%86%85%E5%AE%B9%EF%BC%8C%E4%BB%A5%E4%B8%8B%E8%BF%B0%E8%84%9A%E6%9C%AC%E4%B8%BA%E5%9F%BA%E7%A1%80%EF%BC%8C%E5%AE%8C%E6%88%90%E5%AF%B9%E6%9C%AC%E4%BD%9C%E4%B8%9A%E6%8F%90%E4%BE%9B%E7%9A%84%E3%80%8A%E5%85%A8%E5%AE%8B%E8%AF%8D%E3%80%8B%E7%9A%84embedding)

<https://www.github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/examples/tutorials/word2vec/word2vec_basic.py>

**参考[word2vec\_basic.py](https://www.github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/examples/tutorials/word2vec/word2vec_basic.py)，新建[word2vec\_homework.py](https://www.github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/examples/tutorials/word2vec/word2vec_basic.py)**

### 将全宋词资每个字符作为一个word

|  |
| --- |
| filename ="F:\code\CNN\quiz-w10-code\QuanSongCi.txt"  # Read the data into a list of strings.  def read\_data(filename):  txt\_data = []  with tf.gfile.GFile(filename) as fid:  lines = fid.readlines()  for line in lines:  for s in range(len(line.strip('\n'))):  if s!="" :  txt\_data.append(line[s])  return txt\_data  vocabulary = read\_data(filename) |

### 2.全宋词全文共6010个不同的单字符，这里只取出现次数最多的前5000个单字符。

count = [['UNK', -1]]

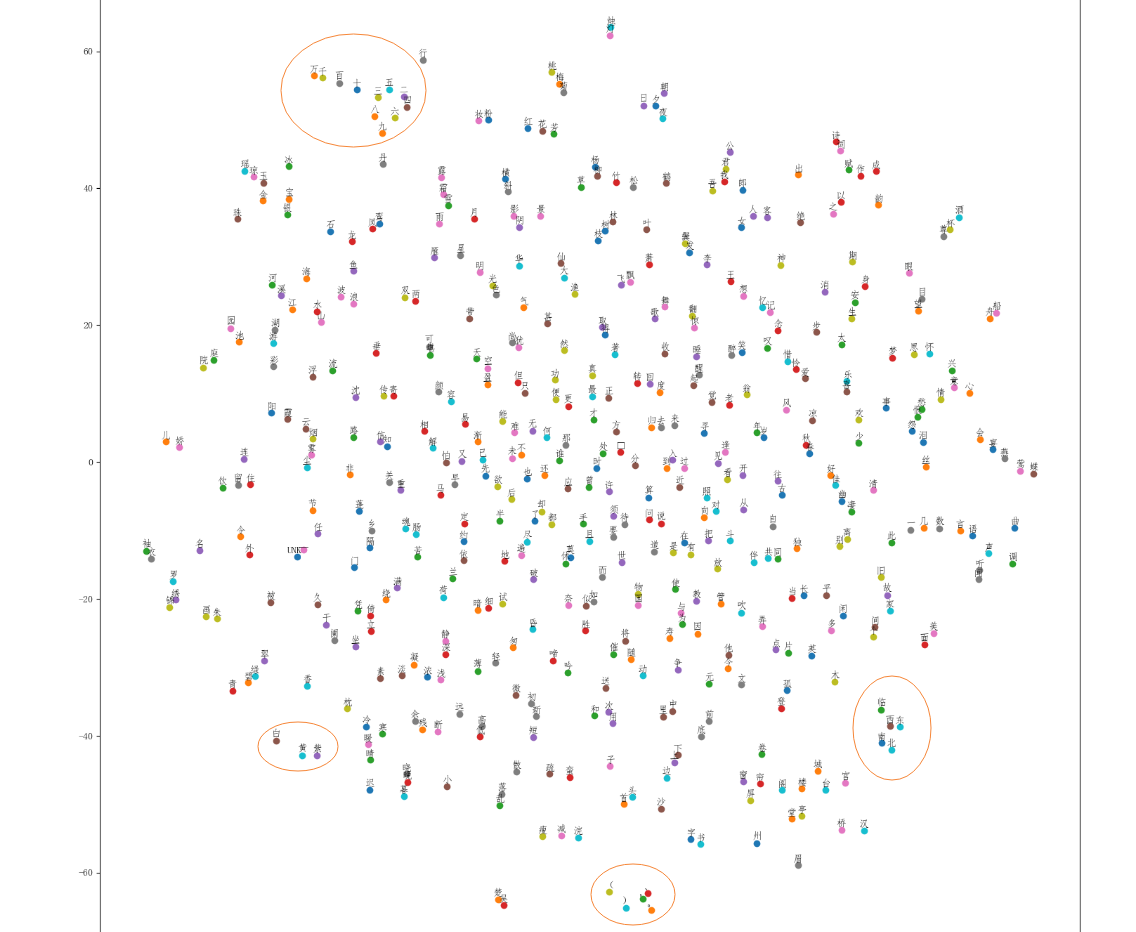
count.extend(collections.Counter(words).most\_common(n\_words - 1))

### 生成文件

|  |
| --- |
| # word2vec中，可以使用如下代码来保存最终生成的embeding np.save('embedding.npy', final\_embeddings)  #保存dictionary dictionary  f1 = open('dictionary.json', 'w')  f1.write(json.dumps(dictionary))  f2 = open('reverse\_dictionary.json', 'w')  f2.write(json.dumps(reversed\_dictionary))  f1.close()  f2.close() |

### 运行40W个step

生成qsc.png



从图片可以看出

数字 二三四五六七八九十百千万

颜色 白紫黄

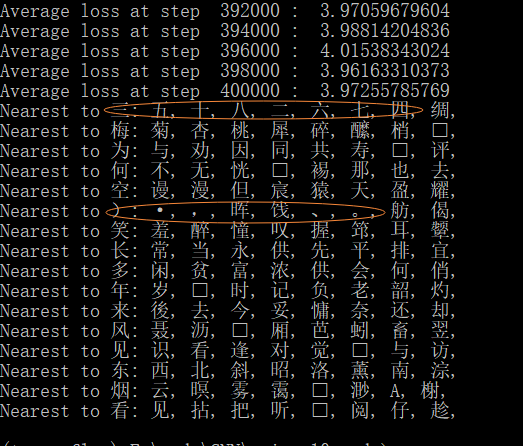
方向 东南西北

标点符号 ( ) . 。：，、等

之间的相似度被训练出来了

### 5.log日志

可以看出模型训练出了词之间的相关性

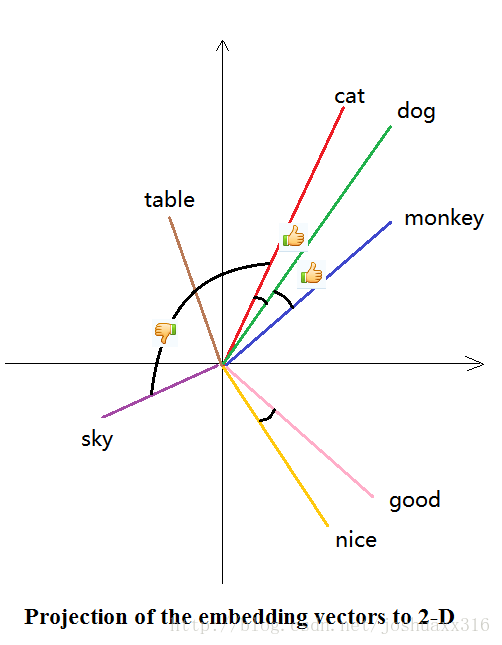


一些词义最相近的八个字还是很对的

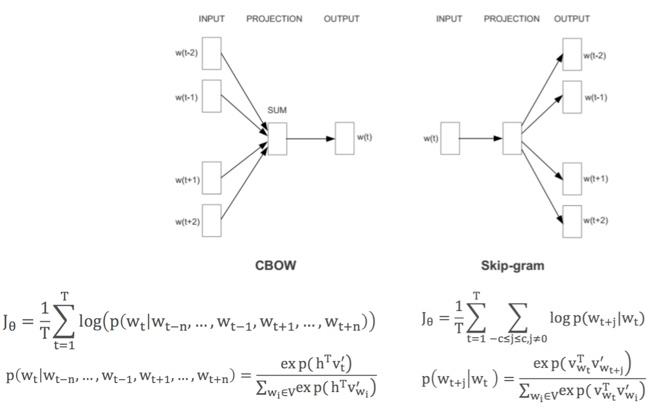
### 6.embedding的理解

单词嵌入

word嵌入是一种分布式的特征表述，向量的不同维度用来表征不同特征，不同维度上就代表着不同的语义。例如苹果和红旗都是红色的，苹果和香蕉都是水果。所以编码中编入的信息包含颜色和属性信息的话，则实质上往不同空间维度上投影，则语义相似性度量的结果是不一样的。



**连续词袋模型（CBOW），Skip-Gram模型**



**6.结果可视化**

**遇到的问题整理：**

Q1:执行[word2vec\_homework.py](https://www.github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/examples/tutorials/word2vec/word2vec_basic.py) 出现

Traceback (most recent call last):

File "word2vec\_homework.py", line 238, in <module>

close\_word = reverse\_dictionary[nearest[k]]

KeyError: 20668

A1：这里只取出现次数最多的前5000个单字符。

Q2：Traceback (most recent call last):

File "word2vec\_homework.py", line 215, in <module>

batch\_size, num\_skips, skip\_window)

File "word2vec\_homework.py", line 124, in generate\_batch

buffer[:] = data[:span]

TypeError: sequence index must be integer, not 'slice'

A2：googel真是神器，这问题也有人遇到，完美解决，

是python版本的问题

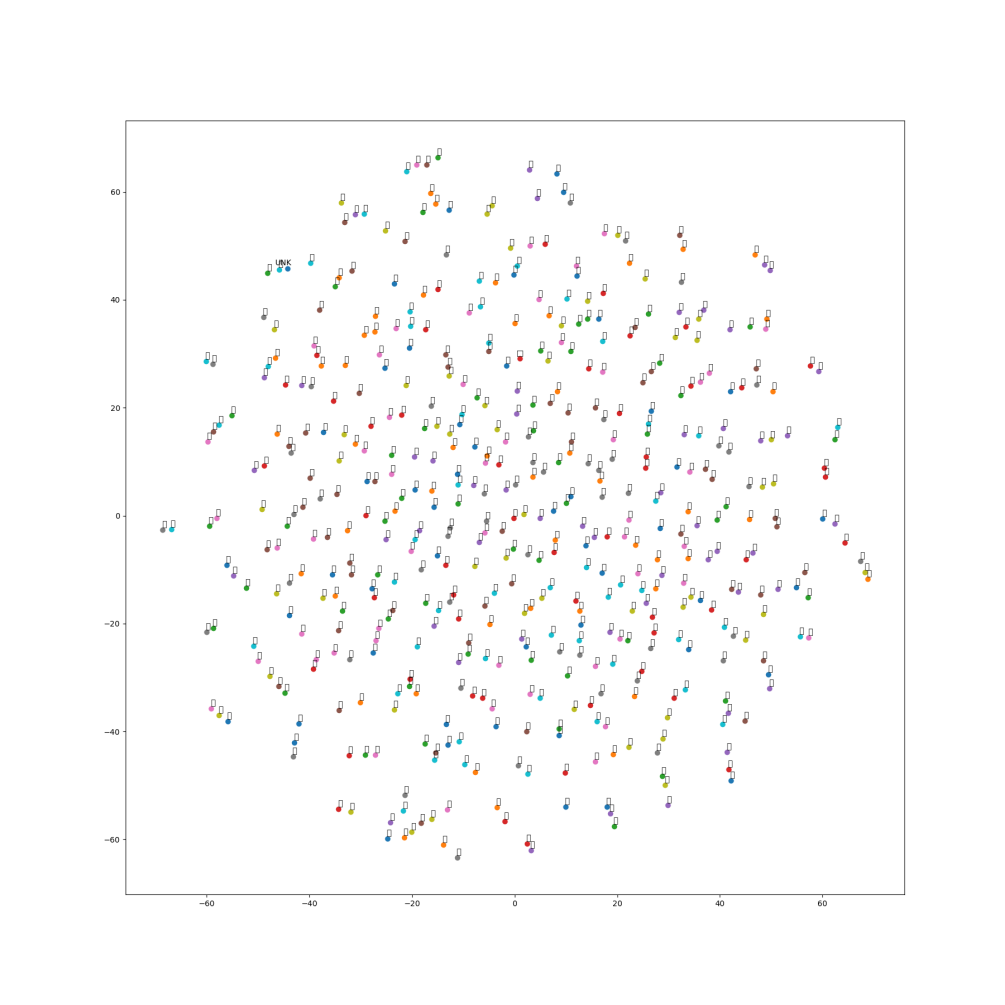
<https://github.com/tensorflow/tensorflow/issues/10866>

buffer[:] = data[:span]改为

for word in data[:span]:

buffer.append(word)

Q3：matplotlib中输出中文的时候出现了乱码



A3:

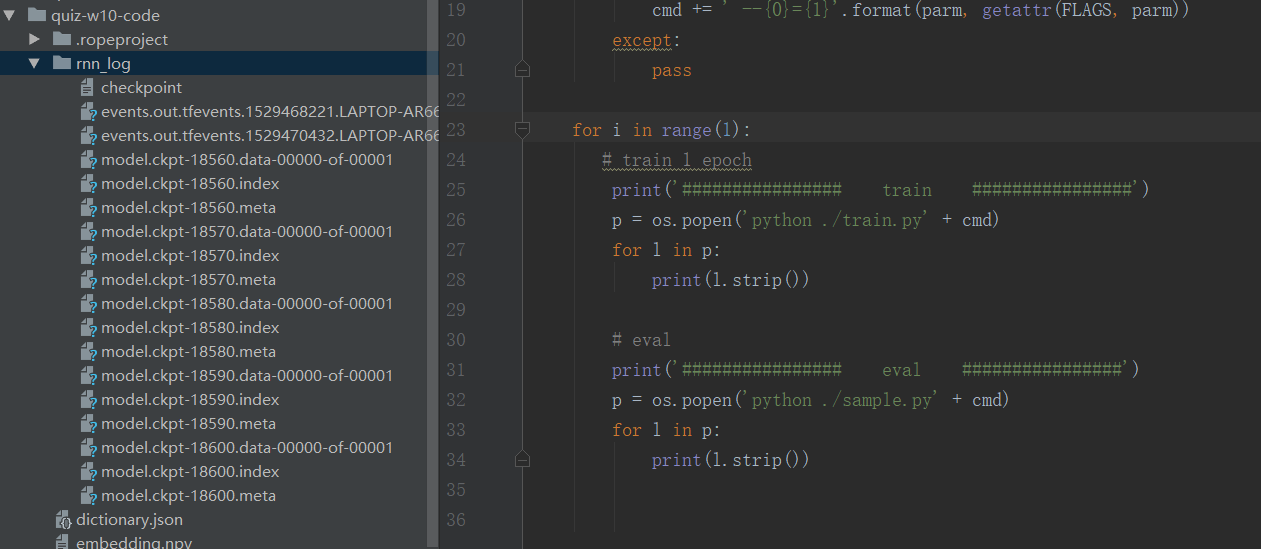
添加下面设置即可

|  |
| --- |
| from pylab import mpl    mpl.rcParams['font.sans-serif'] = ['FangSong'] # 指定默认字体  mpl.rcParams['axes.unicode\_minus'] = False # 解决保存图像是负号'-'显示为方块的问题 |

**rnn训练部分**

先本地循环一次看看效果，

### 1.生成checkpoint文件



### 2.log日志



输出结果和作业上的类似

可以明确看到，RNN学会了标点的使用，记住了一些词牌的名字。

### 代码分析

具体代码请看train.py,model.py,utils.py

model.py

1. 定义使用LSTM结构为循环体且使用dropout的深层循环神经网络

|  |
| --- |
| lstm\_cell = tf.nn.rnn\_cell.BasicLSTMCell(self.dim\_embedding)  if is\_training:  lstm\_cell = tf.nn.rnn\_cell.DropoutWrapper(lstm\_cell, output\_keep\_prob=self.keep\_prob)  cell = tf.nn.rnn\_cell.MultiRNNCell([lstm\_cell] \* self.rnn\_layers) |

2.从LSTM中得到的输出再经过一个全连接层得到最后的预测结构，最终的预测结果在每一时刻上都是一个长度为

num\_words的数据，经过softmax层之后表示下一个位置是不同单词的概率

|  |
| --- |
| with tf.variable\_scope('softmax'):  ##################  weight= tf.get\_variable("weight",[self.dim\_embedding,self.num\_words],initializer=tf.random\_normal\_initializer(stddev=0.01))  bias= tf.get\_variable("bias",[self.num\_words],initializer=tf.random\_normal\_initializer(stddev=0.01))  logits = tf.matmul(seq\_output\_final,weight) + bias  ################## |

utils.py

1. 读取数据，

|  |
| --- |
| def get\_train\_data(vocabulary, dictionary,batch\_size, num\_steps):  ##################  # Your Code here  data\_len = len(vocabulary)  batch\_len = data\_len // batch\_size    raw\_x = [dictionary.get(w, 0) for w in vocabulary]  raw\_y = [dictionary.get(w, 0) for w in vocabulary[1:]]  raw\_y.append(5000 - 1)    batch\_len = data\_len // batch\_size  data\_x = np.zeros([batch\_size, batch\_len], dtype=np.int32)  data\_y = np.zeros([batch\_size, batch\_len], dtype=np.int32)    for i in range(batch\_size):  data\_x[i] = raw\_x[batch\_len \* i: batch\_len \* (i + 1)]  data\_y[i] = raw\_y[batch\_len \* i: batch\_len \* (i + 1)]    epoch\_size = batch\_len // num\_steps  print("epoch\_size,",epoch\_size)    for i in range(epoch\_size):  x = data\_x[:, num\_steps \* i: num\_steps \* (i + 1)]  y = data\_y[:, num\_steps \* i: num\_steps \* (i + 1)]  yield (x, y)  ################## |

train.py

1. 循环的读取上部生成的数据

|  |
| --- |
| for (X,Y) in utils.get\_train\_data(vocabulary,dictionary, batch\_size=FLAGS.batch\_size, num\_steps=FLAGS.num\_steps):  ##################  feed\_dict = {model.X: X,  model.Y:Y,  model.state\_tensor: state,  model.keep\_prob: 1.0} |

LSTM原理：

参考<https://www.jianshu.com/p/dcec3f07d3b5>，写得确实很好

## tinymind运行

本地运行正常代码应该没问题，现在tinymind跑30epoch

<https://www.tinymind.com/executions/k5inttm2>

但tinymind好像有问题，一直处于等待中状态，不知道要等到啥时候，

所以先把作业提交了