## 一、作业一

1、对embedding的理解

<1> one -hot向量和embedding的比较

1. 计算量

英文有26个字母，但独立的一个字母无法表达语义，由字母组成的单词数据量巨大。中文一个单词可以表示相应的语义，单词量也比较大。任何国家的语言，想要完成基本的表达能力，都需要一定数目的单词，这都会导致数据量巨大。所以原来的one-hot不再适用，而embedding这种可以高维度连续表示的方式极大的降低了数量的数目，举个例子二维的平面如果采用one-hot的方式仅仅可以表示俩个值，而embedding可以表示不可数个值。

1. 词之间的关联性无法体现

有些词彼此间有相关性，one-hot的方式是词与词彼此是独立的，但现实生活词语词彼此是依赖的例如学生、和同学等。而embedding这种连续的方式能很好的表达词语词之间的关联性。

<2> 代码实现

这个作业参考word2vec\_basic.py，当通过查找网上例子、打log、自己写一些小栗子，很容易将word2vec\_basic.py解析明白。

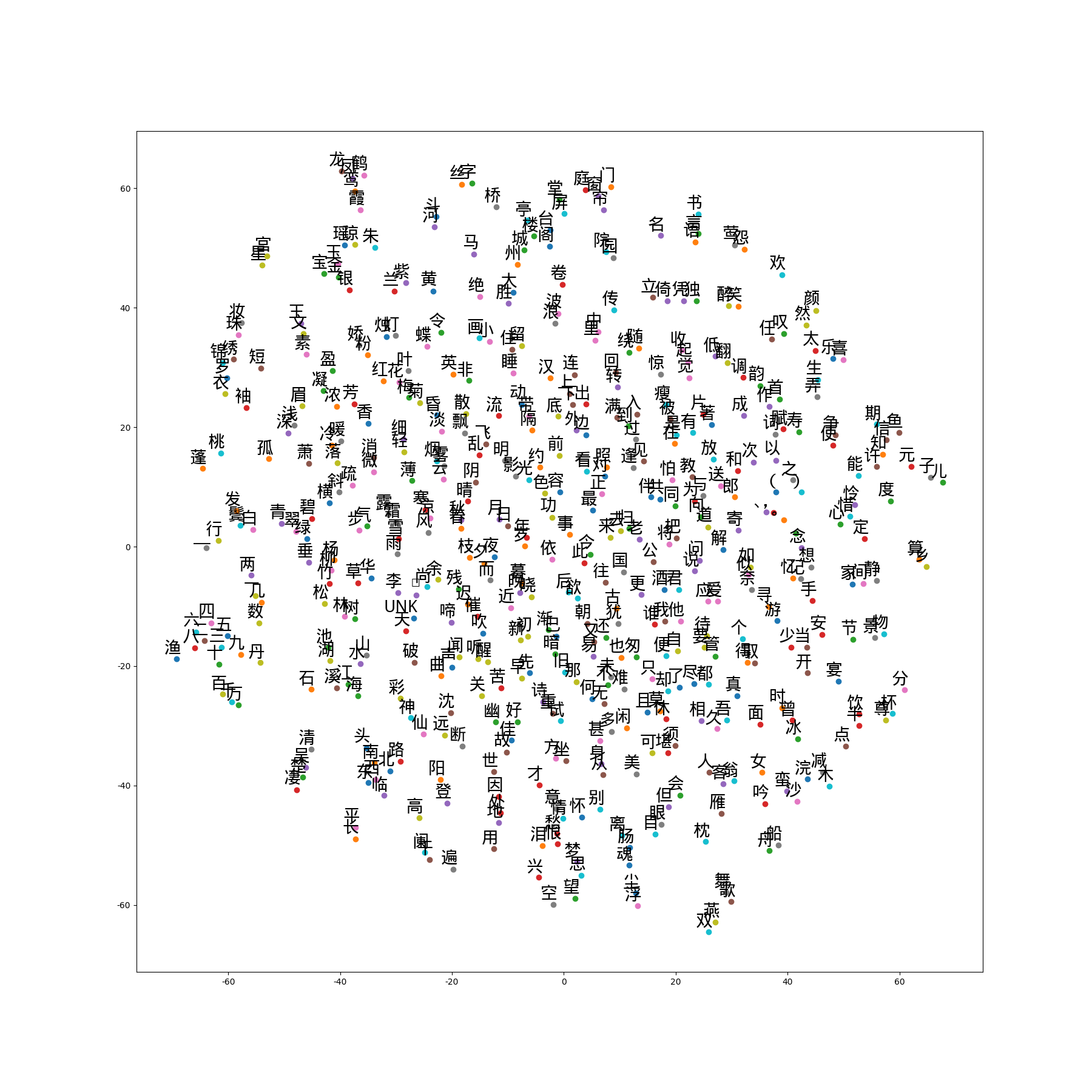
要点如下：

1、将vocabulary\_size = 5000

2、增加中文的设置

3、collections可以实现排序

2、图片的结果分析和认识



总体来说，结果很神奇，很不可思议

举例1：意思相近的举例很近，

比如说，东南西北，亭台楼阁，青翠碧绿，露霜血雨

举例2：标点符号同样距离很近，例如，、。

补充：老师在视频讲解中提到MAN -WOMAN = KING - QUEEN = UNCLE - AUNT由于采取强行将128维降维到二维的这种方式，所以这种关系不是很好的看出来。



## 作业二

1. 对rnn的理解
2. 前向运算和反向传播

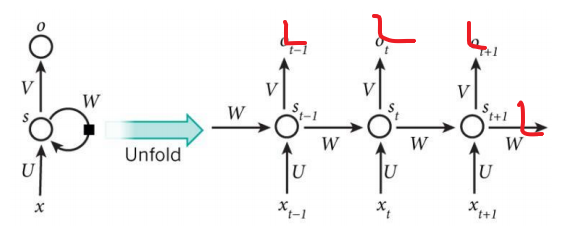
前向运算：略，

反向传播掌握要点如下:

备注：其他则可参考第六周的反向传播

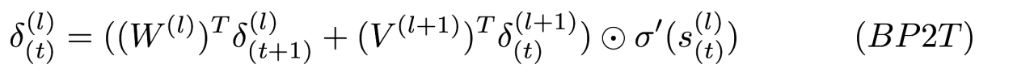
需要从俩个方向传播，沿着时间和沿着输出结果

最外层的layer为每个输出结果以及最后一个state（具体如下图的红色L）



b、梯度消失和梯度爆炸

RNN具有强大的时序记忆能力，但是由于由于vanishing and exploiting gradient问题导致BPTT算法学习不了长期记忆。



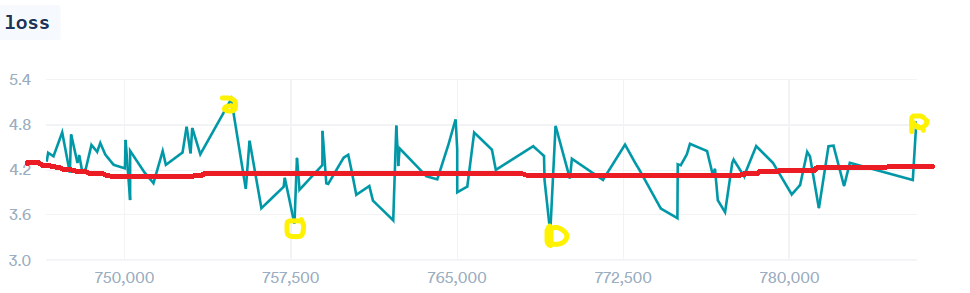
其中当W的特征值大于1的话如果rnn在时间上的维度过深那么由于W相同，则会由于连续乘以W导致梯度爆炸，同理如果W的特征值小于1则会梯度消失。所以为了解决该问题，lstm由此诞生。

1. 训练rnn的过程中的心得体会

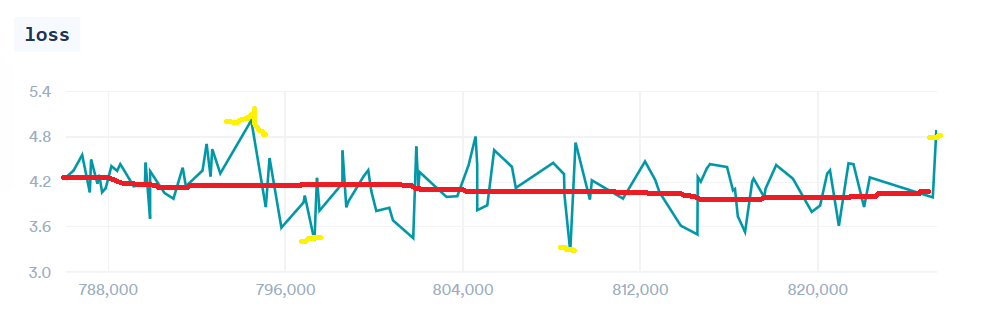
备注：所有的模型都是从上一个模型restore进行训练的

【1】batch size

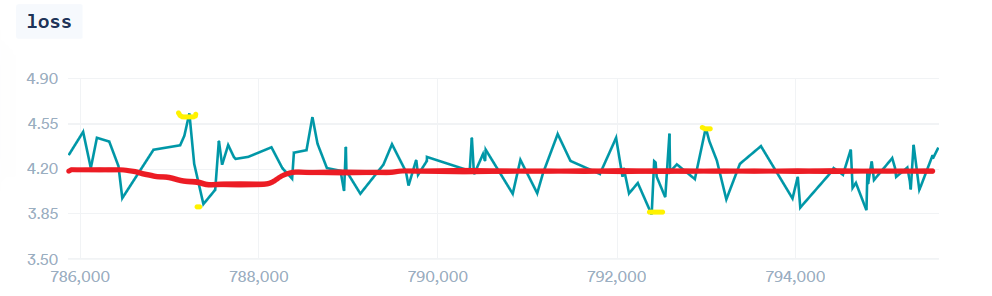
在keep\_prob设置为1的情况下进行调参，通过大概20w个step，loss很好的降到了4.2左右，之后下降就不明显了，其大概沿着红线震荡，也会有一些值超过4.8，也有一些值小于3.6。



尝试改变batch size (由3改为6)，效果不明显，除了训练速度加快些，loss改变不大。



一下是batch size：24.

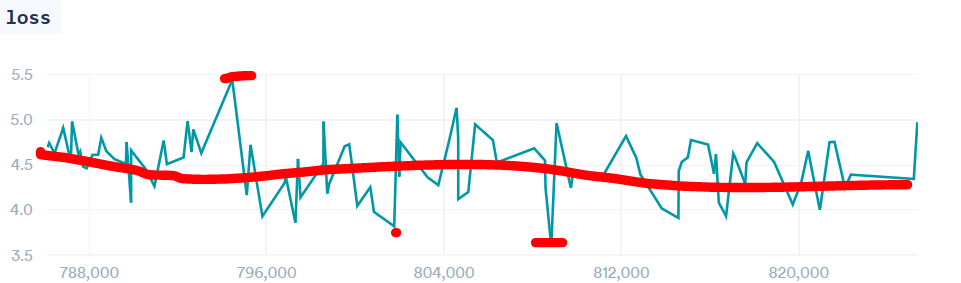


感觉bach size震荡幅度小一些但基本依然是围绕着24。所以感觉更改batch size可能意义并不大。但运行速度有质的飞跃，这个和理论也相符。

一个Epoch 的Step num = data size/batch size/time step.

【2】keep\_prob

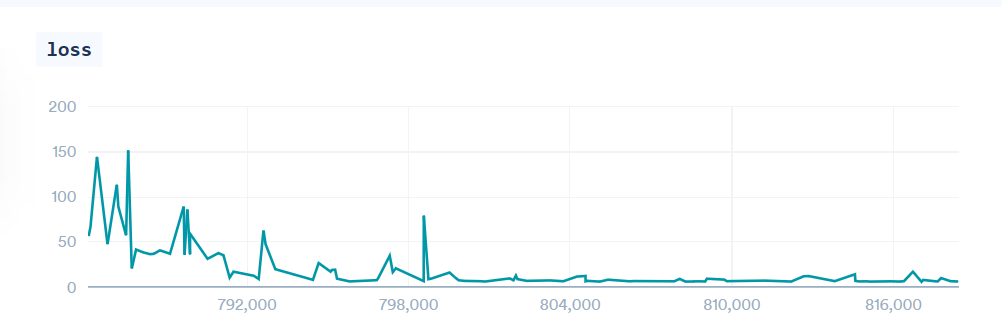
尝试调整keep\_prob，将keep\_prob由1改为0.9，明显开始震荡，效果如下。



感觉调整keep\_prob得不到很好的效果，在其他的神经源网络，调整keep\_prob可以防止过拟合进而改善效果，但是rnn明显不是这个样子，但我没有在网上找到相应的说明，只看到了一些api以及源码，其相关的api如下：



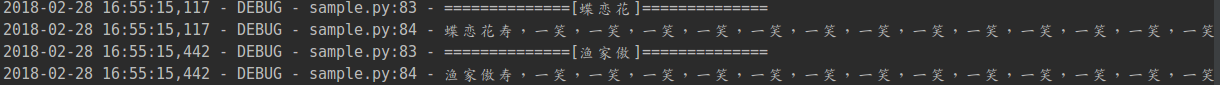
很尴尬，没有找到网上对以上几个api的看法，只有它们的应用例子。下图是将keep\_prob改为0.01会发现原来已经调好的模型立马开始了无脑震荡。

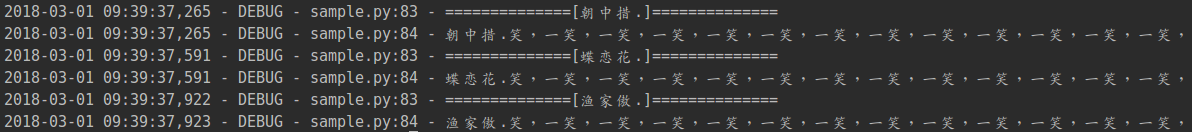


这次作业的参数不是很好调，安装default值执行大概20w个step差不多就达到loss = 4.2左右，如果想进一步降低loss比较困难。

1. 输出的结果的理解以及输出的解释

<1> 输出结果展示





<2>它到底学会了什么

从上图可以看出，机器明显学会了将题目重复一遍以及逗号的使用，“一笑”这个词在文中出现了576次，所以它可能学到了QuanSongCi会高频使用该词。

<3>它为什么会学到这些

首先当loss徘徊在4.2左右的时候，这个时候restore那些模型，然后python3 sample.py会发现仍会有很多回车和句号出现，较好的结果会出现上图的展示，于是我用代码统计了QuanSongCi出现词的频率如下：

('。', 149620), ('\n', 117070)('，', 108451), ('、', 19612), ('人', 13607), ('花', 12809), ('风', 12568), ('一', 11301)

其中'笑'出现了 3425，但是“一笑”出现的概率相对又高一些576次。

举个例子：有一张100分的卷子100道选择题，正确答案为A的数目为42，正确答案为B的数目为42，其余答案的数目为(100-84)个，那么我不看问题全答A或者B都可以拿到42分的成绩。

同上面的例子，机器可能不知道怎么写诗，但他明显学会了这份卷子我都答“。”或者“\n”或者出现概率较高的词都可以有效的降低loss，这可能也是loss在降到一定程度很难降低的原因。