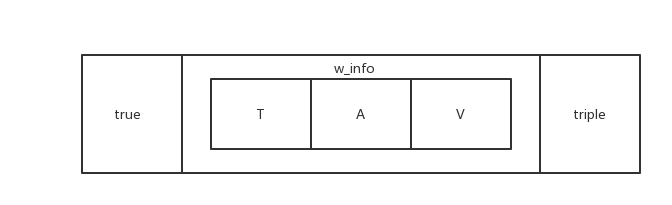
基于已有规则的一点探索

本组已有的方案是完全基于规则的匹配，而这个方案使用的规则十分简洁，并且效果非常出众，因此不禁让人思考，倘若将类似的结构用学习的方法进行处理，是不是能得到更好的效果呢？于是有了如下的探索。至于先前提到的基于规则的匹配，详情请见其文档。

首先尝试分析一下现有规则的特点。在规则匹配时，目标词被分为了3类：T(time)，A(attribute)，V(value)，他们被组合成为一个三元组，他们的具体内容其实是不被关心的，关键的是其相对位置。至于句子中大部分其他词汇，人类在阅读时其实基本忽略了，它们并不影响三元组最终匹配的结果。至此，不禁产生了一个大胆的想法，即完全抛弃句子的实际内容，仅仅根据词的类型和相对位置来学习，让其产生三元组的匹配。由于现有的这个规则实际上十分简洁，但是效果不俗，因此也许仅仅根据这些关键信息就能进行有效的匹配。要实现这一点，首先要做的就是要进行预处理，更改输入数据的形式：



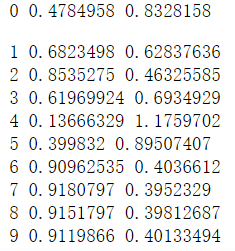
1. 将输入由句子改为三元组。对于每个句子，根据输入的数据，通过三层循环，从times、values、attributes中分别提取元素，匹配出所有可能的三元组，每组作为一个独立的输入数据。对于训练集，所有可能的三元组数量为12万左右（126208），相当之多。
2. 处理输入的信息（为上图中所表示的形式）。
   1. 第一部分为true，数据的分类只有两类：是我们所需的三元组，不是所需的三元组。根据results中的结果，给每个三元组做标注，标记其是否为目标三元组，如果是，则true[1]=1,ture[0]=0；否则true[1]=0,ture[0]=1。
   2. 第二部分为w\_info，它是输入训练的信息。我们需要的仅仅是词的类型以及其位置信息。其中，位置信息关注的并不是绝对位置（它在句中的实际位置），而是其相对于其他目标值的位置，即相对位置。因此，根据原句提取这些信息：以T（time）为例，提供一个句子内T的总数和此三元组中的T为句子中的第几个T。信息的组织形式相当于一个15\*15的稀疏矩阵（稀疏的0,1矩阵），有效信息只存在于其中的某一行。有效信息的纵坐标为一个句子内T的总数，横坐标为此三元组中的T为句子中的第几个T，有效信息的数值为1，其余均为0。假如一个略去所有其他词的句子为,，从中提取出的一个三元组为；那么这个三元组中，关于T的输入为：

对于A与V也如此处理（此例子为3\*3的矩阵，实际上处理为15\*15），并将矩阵压至一维作为输入。

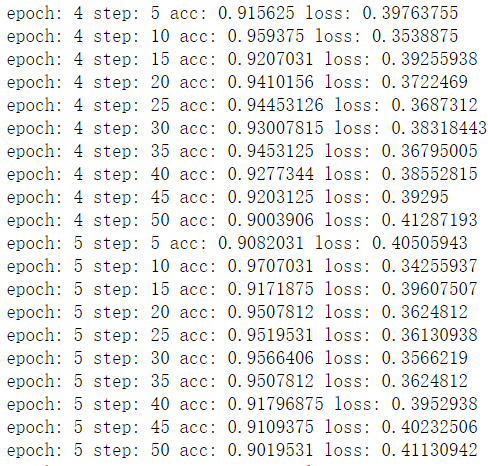
* 1. 第三部分为triple。其中存储的就是T,A,V在原句中的位置，做debug用。

1. 在网络方面，首先尝试一个简单的矩阵相乘，用以熟悉tensorflow，结果果然处理地过于简单，并没有什么效果，但是验证了程序可以运行。随后尝试使用LSTM网络，但是在完成一版代码后，由于个人水平有限（只是阅读一些材料），难以调通，随后决定循序渐进先使用普通的RNN网络。

第一次的尝试运行的结果如下（epoch accuracy lost）：（注意，此处的acc不是以sentence为总量的准确率，而是以所有三元组为总量的准确率）

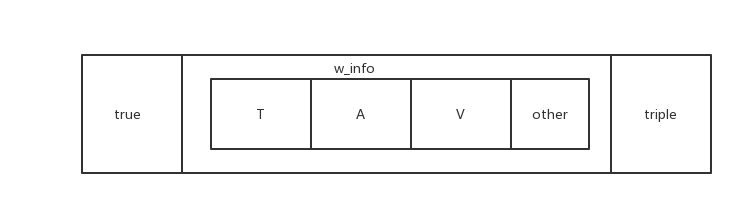


可以看到，结果的震荡过大，查阅后发现自己写的学习率居然达到了0.1，改成0.01或0.001等较小的学习率后，后期的大幅震荡便消失了。另一个问题是，运行的速度很慢，而且学习收敛的速度很低，一番查阅后，发现自己训练时的batch\_size不太合理，一开始由于没有专门写数据输入函数（向占位符传输数据的函数），每次的输入为所有数据，即每次训练都是使用了所有样本，这样造成的后果是，学习次数不够，且运行速度很慢。完成数据输入函数后，每次输入2560组数据，重新运行，结果如下：

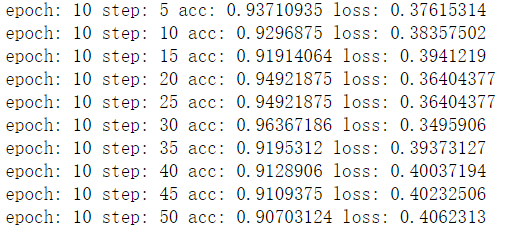


在运行至第5轮时，已经基本稳定，但是由于此前提到，此处的acc不是以sentence为总量的准确率，而是以所有三元组为总量的准确率，而三元组的总数有12万之多，sentence却只有3000句，因此若不能使acc达到至少0.99，这个模型基本没有意义，也不可能超过之前的纯规则匹配。因为即使错误率只有0.01，也会产生1000个三元组判断错误，那么，考虑这1000个错误的三元组分摊到3000个句子中，产生的错误之多是难以承受的，绝对会使得准确率低于0.99，也就难以超过原先的规则匹配了。

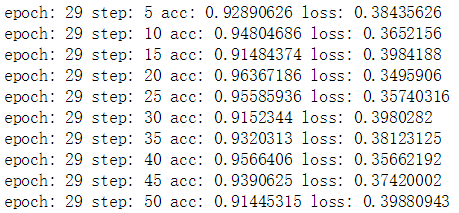
至此，不禁开始思考，是不是现有的模型考虑地过于简单了，因此尝试提供更多的信息，



在others中填入以下信息：所在句子中T、A、V的总数，当前三元组中T、A、V为句子中的第几个T、A、V，当前三元组中T、A、V在句子中的绝对位置。重新训练，结果如下：



与之前对比，发现并没有什么提升，总准确率仍然在0.92上下，离目标0.99+相去甚远，不禁开始怀疑，整个思路也许出现了问题。事实上后来的验证也发现，这个思路的确存在硬伤。更加激进的处理输入数据，将w\_info改为仅仅20位，包含的信息如下：所在句子中T、A、V的总数；当前三元组中T、A、V为句子中的第几个T、A、V；当前三元组中T、A、V在句子中的绝对位置；当前三元组中T、A的绝对位置差，当前三元组中A、V的绝对位置差；原句中第1、2、3、4、5个逗号的绝对位置（如果有的话）；第1个分号的绝对位置（如果有的话），第一个逗号与当前三元组中T的位置差；第一个逗号与当前三元组中A的位置差。将如此少的信息输入训练，结果如下：



由于训练输入如此之少，训练可以进行地相当之快，然而直至29轮，准确率仍然只是在0.92左右徘徊，与之前相比毫无长进，可见，这个模型下，无论如何调参，较为简单的判断是正确的，较为困难的判断则似乎无法通过当前所给的信息进行判断，因此不得不承认这是一个不太成功的探索。根据另一位组员所做的相关的尝试，在以三元组为总数判断准确率达到0.93左右的情况下，recall率为0.17，prec为0.62，correct为0.15，F1为0.27可以说是相当之惨



虽然我们实现的细节和具体框架不同，但是可以推知，本模型的效果也无法超过此结果。倘若不能使得以三元组为总数判断准确率达到0.99+，此模型基本没有使用价值。

尝试分析失败原因：

1. 模型的设置尝试模拟本组规则匹配的思考过程，极大简化输入数据，但是这一切的出发点忽略了人类思维的灵活性，在使用此规则匹配时，人类实际上在不经意间进行了很多判断，利用了很多此模型忽略的信息，导致这个模型的输入实际上无法进行有效的判断。
2. 模型采用的输入是以三元组为单位的，而不是以句子为单位；acc与lost的计算也是基于三元组，而不是sentence，这会导致训练目标与实际目标脱离，不能有效的契合此实验的目标。
3. 由于以三元组为单位，此模型大概不能很好判断三元组数量与句子的关系，即一个句子应该产出多少个三元组，弱化了三元组与原句本身的联系，导致丢失了很多必要的信息。