模型设计

由于绘图并不能很好地展示网络结构的细节，并且整体上我的 CNN 和 RNN 同文档上的方案相同，所以这里只贴出文档上的结构图，并且**给出了网络结构信息。**

中文词向量处理

使用预训练的词向量，repo：https://github.com/embedding/chinese-word-vectors

由于数据为新闻类，所以使用其中的 Sogou News 的 SGNS (Word + Character + Ngram) 对应的词向量。

关于 unk 的处理

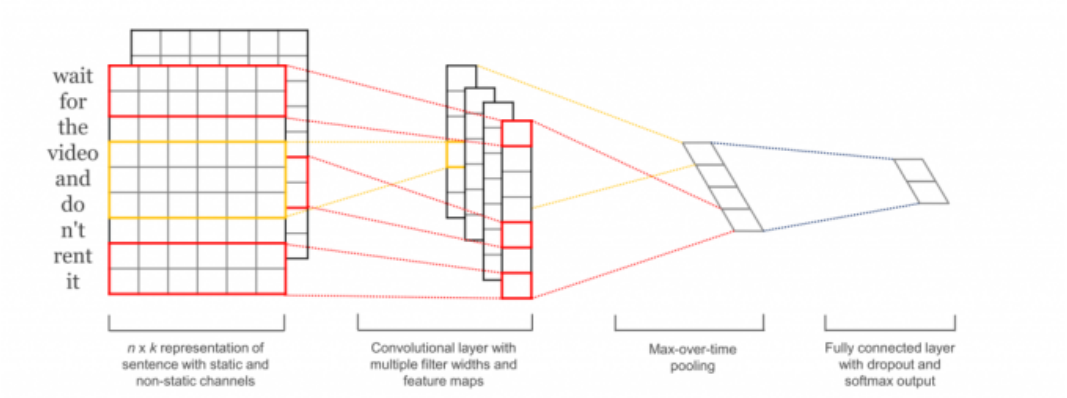
即对于未登录词的处理。

在 issue (https://github.com/Embedding/Chinese-Word-Vectors/issues/54)中发现，该预训练的词向量并没有对 unk 词做词嵌入，而作者的建议是使用整体词向量的均值或者存在下游模型时随机初始化一个 unk 向量。

这里采用随机初始化的方法，固定一个向量作为 unk 向量。

CNN

网络结构



分 3、4、5 三个卷积核（代表不同的观测长度）进行 Conv2d 操作，并且利用 MaxPool2d、Dropout 对这三个结果分别进行池化和 dropout 操作：

```
self.cnn1 = nn.Sequential(
    nn.Conv2d(1, Config.filter_num, kernel_size=(3, Config.vec_len)),
    nn.ReLU(),
    nn.MaxPool2d((Config.seq_len - 3 + 1, 1)),
    nn.Dropout(p=.2),
)
self.cnn2 = nn.Sequential(
    nn.Conv2d(1, Config.filter_num, kernel_size=(4, Config.vec_len)),
    nn.ReLU(),
    nn.MaxPool2d((Config.seq_len - 4 + 1, 1)),
    nn.Dropout(p=.2),
```

```
)
self.cnn3 = nn.Sequential(
    nn.Conv2d(1, Config.filter_num, kernel_size=(5, Config.vec_len)),
    nn.ReLU(),
    nn.MaxPool2d((Config.seq_len - 5 + 1, 1)),
    nn.Dropout(p=.2),
)
```

最终将这三个结果合并之后连入一个全连接层：

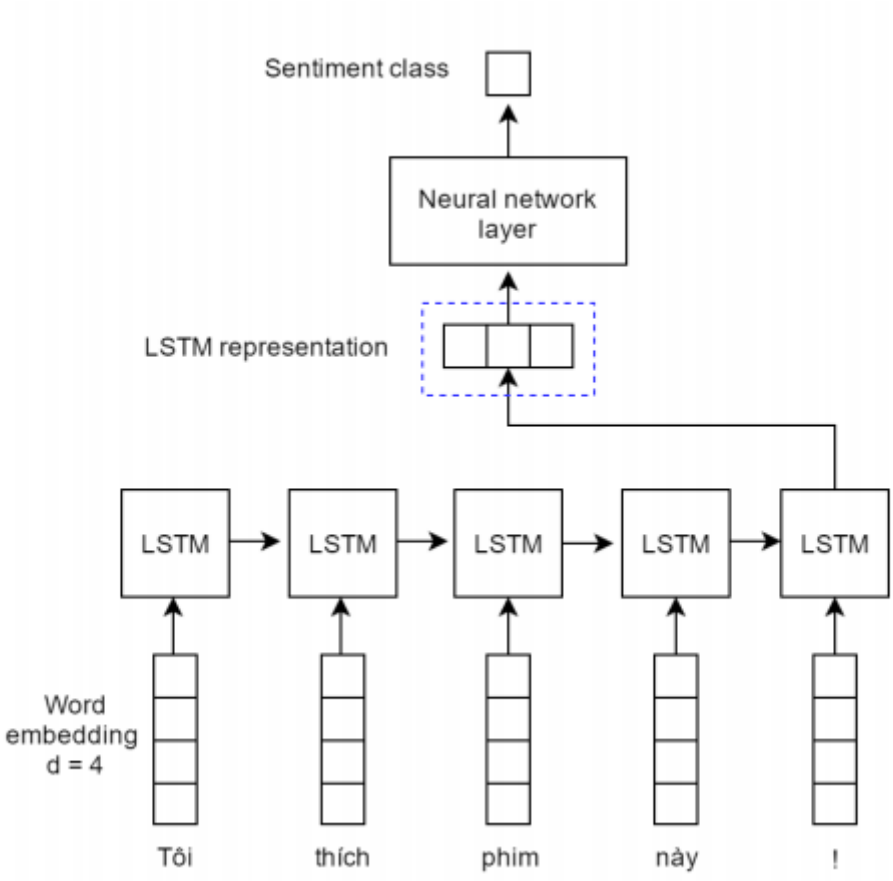
```
self.fc = nn.Linear(3 * Config.filter_num, Config.label_len)
```

参数设置

label_len	vec_len	seq_len	filter_num	train_batch_size	learning rate	momentum
8	300	500	4	32	0.001	0.9

RNN

网络结构



使用单层的 LSTM 网络：

```
self.lstm = nn.LSTM(input_size=Config.vec_len, hidden_size=16,
                                num_layers=1, dropout=0.2)
```

最终进行全连接：

```
self.fc = nn.Linear(16, Config.label_len)
```

参数设置

label_len	vec_len	seq_len	train_batch_size	learning rate	momentum
8	300	500	32	0.001	0.9

实验结果

CNN

```
loss: 1.118  cost: 2.984  train_acc: 0.609  valid_acc: 0.493  test_acc: 0.536
```

测试集准确率到达了 0.536，并且此时训练集准确率为 0.609，过拟合程度还不大。

使用宏平均，f1 score 为 0.16068。

RNN

```
loss: 1.705  cost: 4.066  train_acc: 0.397  valid_acc: 0.417  test_acc: 0.464
```

准确率比起 CNN 来说差了一些，只达到了 0.464。

使用宏平均，f1 score 为 0.07865。

参数比较

- dropout：当 dropout 提高时，能够有效地防止过拟合，具体来说即训练集的准确率提升速度将下降，相应的训练时间也会变长——这是因为随机丢弃的概率增大之后，训练的效率将收到影响。
- learning rate：较小时将导致训练速度慢，但过大容易导致最终得到的准确率更低，即训练效果可能会变差——原因很明显，当 learning rate 很大时，容易 "错过" 更优值而到达更差的参数上。
- 网络层数或单层的神经元数目：越大将导致网络训练越慢，但可能会带来更好的训练效果，然而过大时容易造成过拟合。

问题思考

实验训练的停止时间

需要小心的**防止过拟合**。

在我的实现中，将原本的训练集拆分为了训练集和验证集两部分，通过观察验证集的准确率变化，找到使得验证集准确率最高的时刻停下。

固定迭代次数难以动态适应训练过程，并且没有一个合适的指标判断是否过拟合。而使用验证集，可以方便地找到一个合适的停止时刻，既减缓过拟合，又能不陷入欠拟合。

实验参数的初始化

一般不采用全 0 初始化或者说全为同一个常数的初始化方法。

常见的一些初始化方法包括正态分布初始化、Xavier 初始化、Kaiming 初始化等等。其中正态分布初始化可以应用于一般情况，是较为常见的一种初始化方案；Xavier 初始化可以帮助减少梯度弥散问题，使得信号在神经网络中可以传递得更深入，一般适用于 tanh 和 softsign；Kaiming 初始化适用于，在正向传播时状态值的方差保持不变，在反向传播时激活值梯度的方差保持不变的情况。

防止过拟合

可以考虑加入 dropout 层，加入惩罚项（例如 L1、L2 正则化项）等。

CNN、RNN、MLP 的比较

- CNN、RNN 参数相较于 MLP 来说一般要少很多，而相较而言，CNN 一般又比 RNN 参数少一些；
- CNN 一般适用于提取相邻空间区域的信息，常用于文本、图像领域；RNN 更适用于提取时序信息，所以在文本处理领域较 CNN 来说应用更加广泛；而 MLP 一般可以作为 baseline 用于比较其它网络的性能。