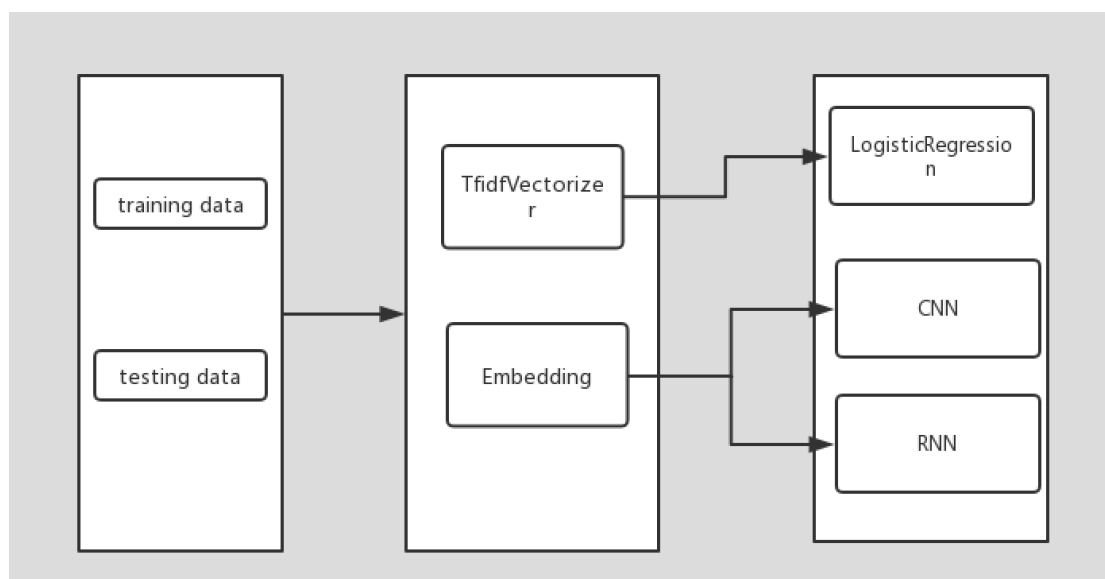


1 模型结构



1.1 LogisticRegression

此模型作为 baseline 模型,逻辑回归函数形式如下：

$$h_{\theta}(x) = g(\theta^T x), g(z) = \frac{1}{1 + e^{-z}}$$

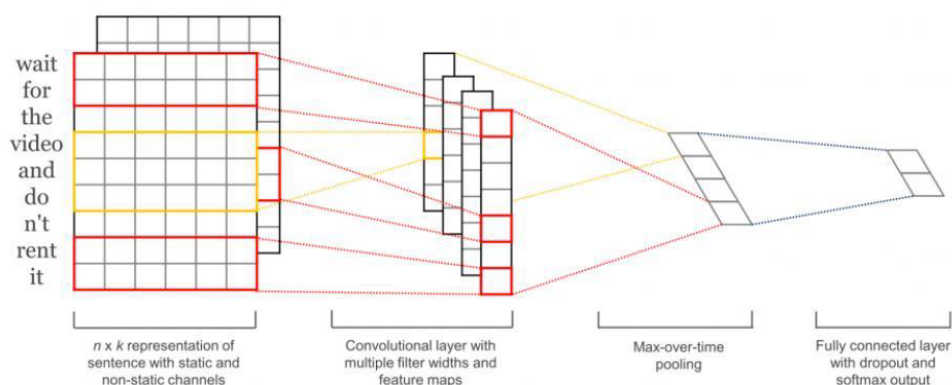
代价函数为：

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m (y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))) \right]$$

其中 y 为 label 值，优化算法采用随机梯度下降算法 SGD。

1.1 CNN

CNN 最初用于处理图像问题，但是在自然语言处理中，使用 CNN 进行文本分类也可以取得不错的效果。在文本中，每个词都可以用一个行向量表示，一句话就可以用一个矩阵来表示，那么处理文本就与处理图像是类似的了。



1.1.1 卷积层

每个卷积核的大小为 $\text{filter_size} \times \text{embedding_size}$ 。filter_size 代表卷积核纵向上包含单词个数，即认为相邻几个词之间有词序关系，代码里使用的是[3, 4, 5]。embedding_size 就是词向量的维数。每个卷积核计算完成之后我们就得到了 1 个列向量，代表着该卷积核从句子中提取出来的特征。有多少个卷积核就能提取出多少种特征，即图中在纵深方向上 channel 的数量。

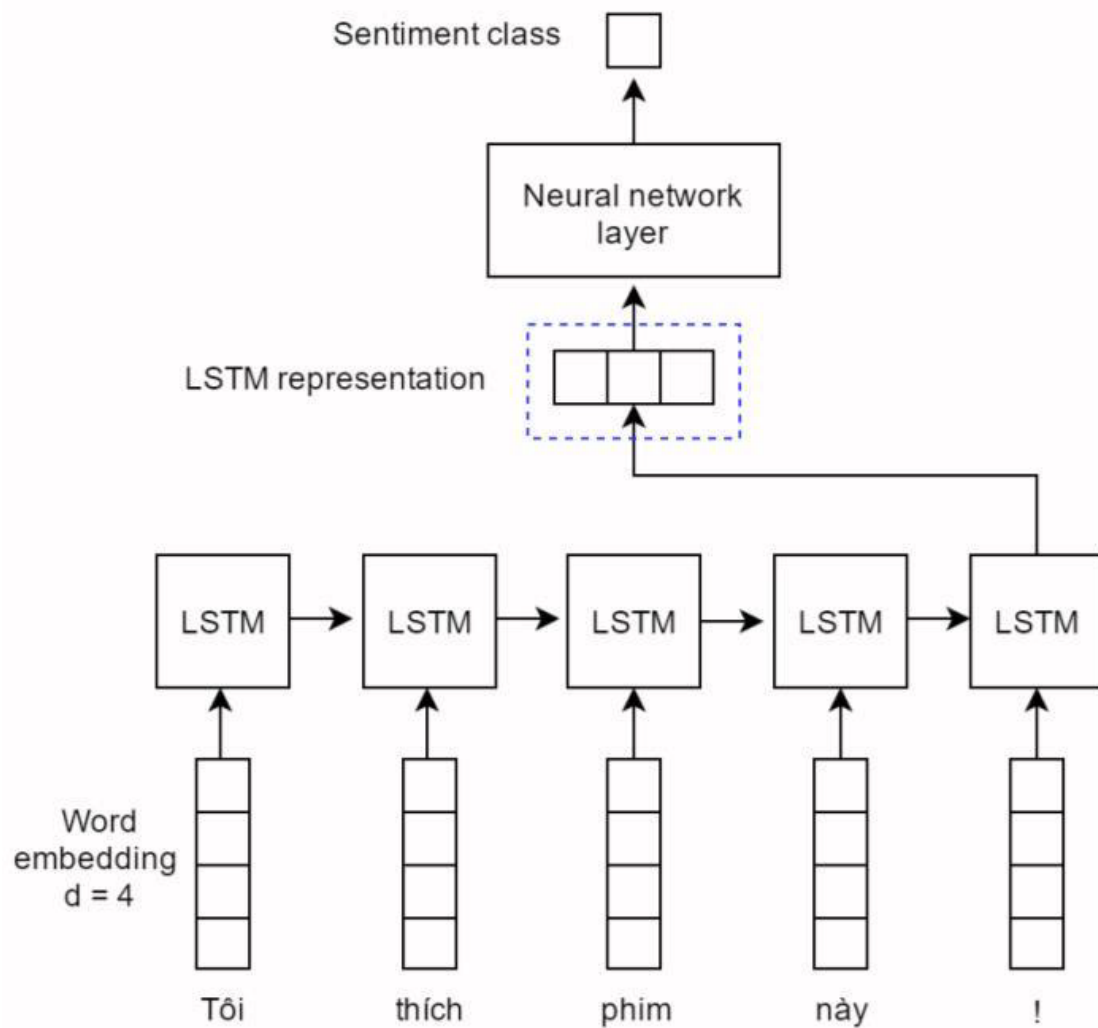
1.1.2 池化层

pooling 操作就是将卷积得到的列向量的最大值提取出来。这样 pooling 操作之后我们会获得一个 num_filters 维的行向量，即将每个卷积核的最大值连接起来。这样做还有一个好处就是，如果我们之前没有对句子进行 padding 操作，那么句子的长度是不同的，卷积之后得到的列向量维度也是不同的，可以通过 pooling 来消除句子之间长度不同的差异。

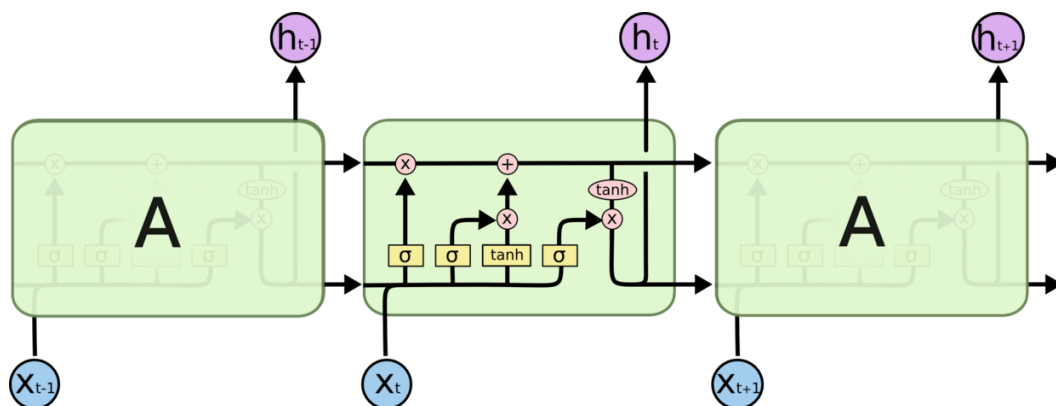
本实验采用了三层卷积神经网络，最终输出层为全连接层，激活函数为 softmax, 输出 8 中类别所对应的概率。

```
sequence_input = Input(shape=(400,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
l_cov1 = Conv1D(128, 5, activation='relu')(embedded_sequences)
l_pool1 = MaxPooling1D(5)(l_cov1)
l_pool1 = Dropout(0.25)(l_pool1)
l_cov2 = Conv1D(64, 5, activation='relu')(l_pool1)
l_pool2 = MaxPooling1D(5)(l_cov2)
l_cov3 = Conv1D(32, 5, activation='relu')(l_pool2)
l_pool3 = MaxPooling1D(3)(l_cov3) # global max pooling
l_flat = Flatten()(l_pool3)
l_dense = Dense(128, activation='relu')(l_flat)
preds = Dense(len(df['label'].value_counts()), activation='softmax')(l_dense)
```

1.2 RNN



本实验采用的 RNN 变体结构 LSTM, LSTM 是一种变种的 RNN, 它的精髓在于引入了细胞状态这样一个概念, 不同于 RNN 只考虑最近的状态, LSTM 的细胞状态会决定哪些状态应该被留下来, 哪些状态应该被遗忘。



LSTM 结构更为复杂，在 RNN 中，将过去的输出和当前的输入 concatenate 到一起，通过 tanh 来控制两者的输出，它只考虑最近时刻的状态。在 RNN 中有两个输入和一个输出。

而 LSTM 为了能记住长期的状态，在 RNN 的基础上增加了一路输入和一路输出，增加的这一路就是细胞状态，也就是途中最上面的一条通路。事实上整个 LSTM 分成了三个部分：

- 1) 哪些细胞状态应该被遗忘
- 2) 哪些新的状态应该被加入
- 3) 根据当前的状态和现在的输入，输出应该是什么

下面来分别讨论：

- 1) 哪些细胞状态应该被遗忘

这部分功能是通过 sigmoid 函数实现的，也就是最左边的通路。根据输入和上一时刻的输出来决定当前细胞状态是否有需要被遗忘的内容。举个例子，如果之前细胞状态中有主语，而输入中又有了主语，那么原来存在的主语就应该被遗忘。concatenate 的输入和上一时刻的输出经过 sigmoid 函数后，越接近于 0 被遗忘的越多，越接近于 1 被遗忘的越少。

- 2) 哪些新的状态应该被加入

继续上面的例子，新进来的主语自然就是应该被加入到细胞状态的内容，同理也是靠 sigmoid 函数来决定应该记住哪些内容。但是值得一提的是，需要被记住的内容并不是直接 concatenate 的输入和上一时刻的输出，还要经过 tanh，这点应该也是和 RNN 保持一致。并且需要注意，此处的 sigmoid 和前一步的 sigmoid 层的 w 和 b 不同，是分别训练的层。细胞状态在忘记了该忘记的，记住了该记住的之后，就可以作为下一时刻的细胞状态输入了。

- 3) 根据当前的状态和现在的输入，输出应该是什么

这是最右侧的通路，也是通过 sigmoid 函数做门，对第二步求得的状态做 tanh 后的结果过滤，从而得到最终的预测结果。

事实上，LSTM 就是在 RNN 的基础上，增加了对过去状态的过滤，从而可以选择哪些状态对当前更有影响，而不是简单的选择最近的状态。LSTM 网络随后被证明比传统的 RNNs 更加有效，尤其当每一个时间步长内有若干层时，整个语音识别系统能够完全一致的将声学转录为字符序列。目前 LSTM 网络或

者相关的门控单元同样用于编码和解码网络，并且在机器翻译中表现良好,一定程度上解决 RNN 梯度消失的问题。

本实验采用的 Bi-LSTM，网络结构参数如下：

```
embedding_matrix = np.random.random((len(word_index) + 1, 100))
embedding_layer = Embedding(len(word_index) + 1,
                             100, weights=[embedding_matrix],
                             input_length=500, trainable=True)

sequence_input = Input(shape=(500,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)

l_lstm = Bidirectional(LSTM(100))(embedded_sequences)
preds = Dense(8, activation='softmax')(l_lstm)
model = Model(sequence_input, preds)
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['acc'])
```

2 实验结果

2.1 模型对比实验

模型	准确率	micro-f1	相关系数
LogisticRegression	24%	0.367	(-0.112267260278899, 1.078143164187129e-07)
CNN	51%	0.509	(0.18324111852688763, 2.8227299145730296e-18)
RNN	37%	0.378	(0.03302518621325931, 0.11913963331472249)

本节实验主要对比了以逻辑回归作为的 baseline 模型与 CNN/RNN 模型效果的差异，可以发现 CNN/RNN 在准确率和 micro-f1 上都有较大幅度的提升。

2.2 参数对比实验

2.2.1 LogisticRegression 参数设置

baseline 模型采用的特征为 tfidf 向量化矩阵，在这里参数对比设置主要为 ngram, N-Gram 是一种基于统计语言模型的算法。它的基本思想是将文本里面的内容按照字节进行大小为 N 的滑动窗口操作，形成了长度是 N 的字节片段序列。比如“我 爱 北京 天安门”，如果采用 2-gram,那么提取到的序列为（我，爱），（爱，北京）。

```
tf_vec = TfidfVectorizer(ngram_range=(1, 2), max_df=0.95, min_df=2)
X = tf_vec.fit_transform(df['news'])
```

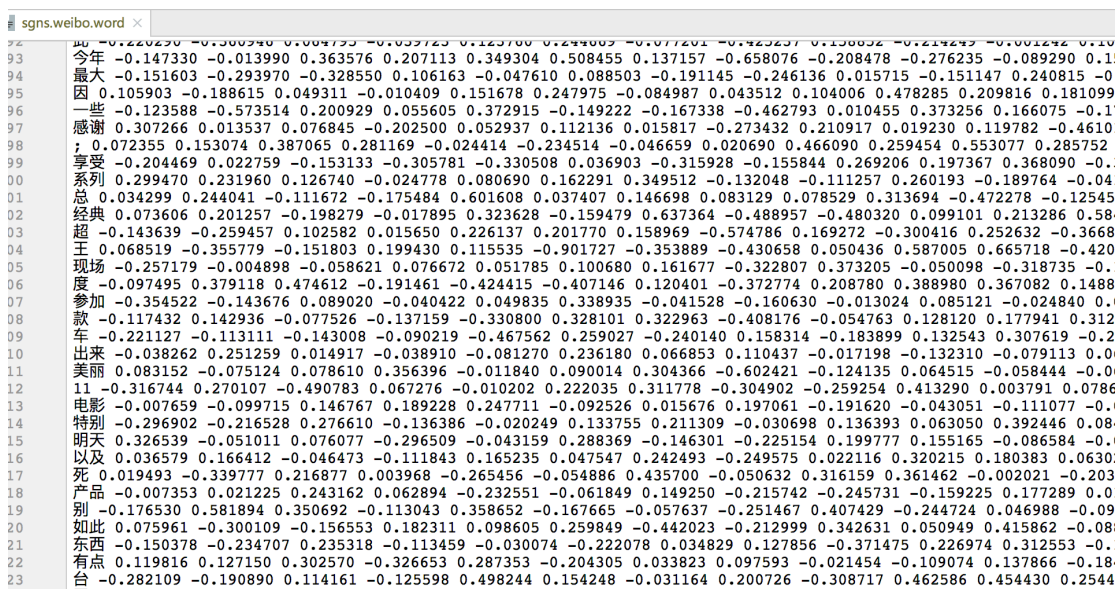
参数	准确率	micro-f1	相关系数
ngram_range=(1, 1)	24%	0.367	(-0.112267260278899, 1.078143164187129e-07)
ngram_range=(1, 2)	22%	0.339	(-0.12152570298283832, 8.698153586740464e-09)

通过上述对比实验可以发现当 ngram 取 (1, 1) 时的效果比较好,这是因为我们数据集的文本是新闻, 长度相比多文本比较大, 并且单个文本出现的词语数量比较多, 像两两词语出现的频率比较少, 对文本的情感影响不大, 可能产生噪音。

2.2.2 有无预训练词向量

因为本实验的数据集为新浪新闻, 所以采用了基于新浪微博语料训练的 300 维 word2vec 词向量, 词向量的下载链接为:

<https://github.com/Embedding/Chinese-Word-Vectors>, 下面为词向量示例:

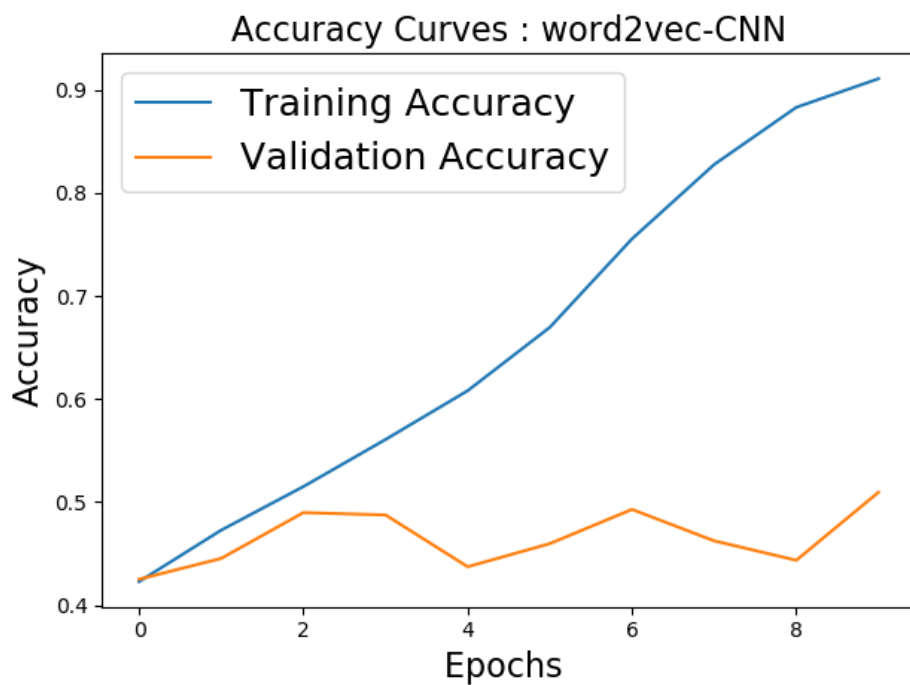
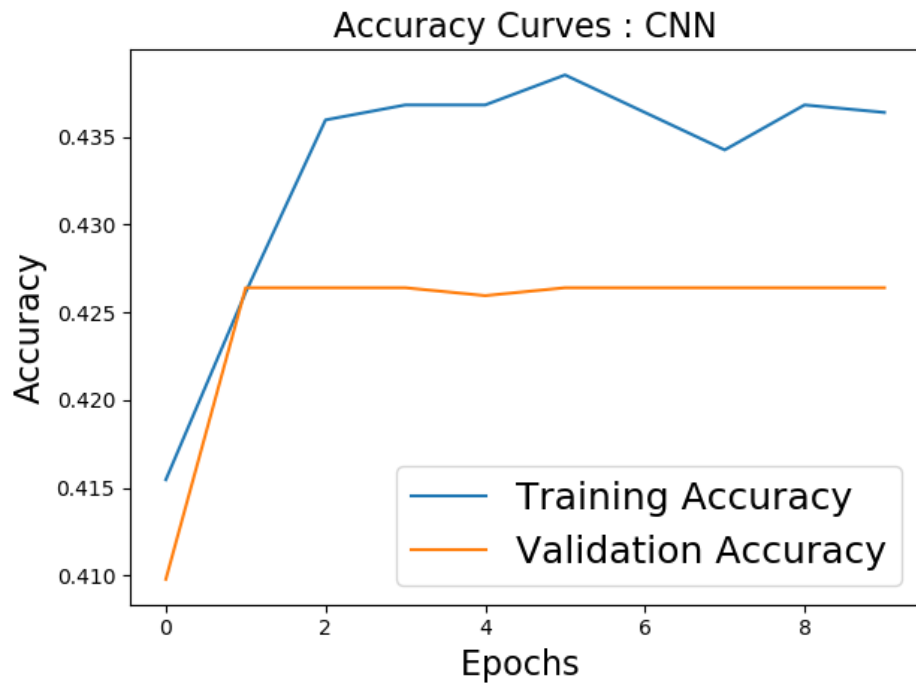


The image shows a screenshot of a text file named 'sgns.weibo.word'. It contains a list of words and their corresponding 300-dimensional word2vec embeddings. The words are listed on the left, and the numerical values of the vectors are listed on the right, separated by spaces. The words include: 今年, 最大, 因, 一些, 感谢, 享受, 系列, 总, 经典, 超, 王, 现场, 度, 参加, 款, 车, 出来, 美丽, 11, 电影, 特别, 明天, 以及, 死, 产品, 别, 如此, 东西, 有点, 台. The numerical values are represented as a long sequence of floating-point numbers.

为了探究是否使用预训练好的词向量对分类效果的影响, 下面以 CNN 的实验对比为例, 下面是实验结果:

参数	准确率	micro-f1	相关系数
----	-----	----------	------

使用 word2vec	51%	0.509	(0.13374504721932695, 2.3285235209304245e-10)
不使用 word2vec	42%	0.426	(1e-10,0.92545545656)



第一张图为没有使用词向量，第二张图使用了词向量，可以发现使用词向量的效果提升比较明显，但是会造成过拟合，这是因为我们训练集比较少，一共仅

有 2300 多条，在训练一定步数之后，会造成在训练集上表现很好，造成过拟合。

2.2.3 其他参数

- 同时在 CNN 和 RNN 的实验过程，也调整了其他参数，比如训练批数据大小 Batch_Size,在不使用预训练词向量的不同 embedding_dim(100,300),发现 b 发现这两个参数可能会影响模型训练的速度，但是最终的评估指标差不多。
- 另外训练长度对训练效果影响比较大，当序列长度过小（小于 100）或者过大时（大于 600）模型训练效果比较差，这是应为序列长度过小时，造成信息损失，不能更好表达文章的语义；当序列长度过大时，会造成矩阵稀疏，模型不收敛。

3 文本思考

3.1 实验训练什么时候停止是最合适的？简要陈述你的实现方式，并试分析固定迭代次数与通过验证集调整等方法的优缺点。

答：在测试误差开始上升之前，就停止训练,即使此时训练尚未收敛(即训练误差未达到最小值)；实现方式是设置训练迭代次数；在学习率适中，迭代次数较大时变量初始化方式对最终准确率的影响不大。

3.2 实验参数的初始化是怎么做的？不同的方法适合哪些地方？

（现有的初始化方法为零均值初始化，高斯分布初始化，正交初始化等）

答：随机初始化。随机初始化可以适用于预训练词向量初始化；零均值初始化和高斯分布初始化适合权重初始化；正交初始化：用以解决深度网络下的梯度消失、梯度爆炸问题，在 RNN 中经常使用的参数初始化方法。

3.3 过拟合是深度学习常见的问题，有什么方法可以方式训练过程陷入过拟合

答：添加 Dropout 层，使用 L1 和 L2,增加训练集、数据增强、简化神经网络、earlystop 等。

3.4 试分析 CNN, RNN, 全连接神经网络 (MLP) 三者的优缺点。

答：CNN 优点为：sparse interaction(稀疏的交互), parameter sharing(参数共享), equivalent representation(等价表示)。适合于自动问答系统中的答案选择模型的训练。CNN 缺点为：需要调参，需要大样本量，训练最好要 GPU；物理含义不明确。

RNN 优点为：分布式表达；能在序列预测中明确地学习和利用背景信息；具有长时间范围内学习和执行数据的复杂转换能力。RNN 缺点为：会造成梯度消失问题；会造成梯度爆炸问题；

MLP 优点为：高度的并行处理；. 2) 高度的非线性全局作用；MLP 缺点为：网络的隐含节点个数选取非常难；停止阈值、学习率、动量常数需要采用“trial-and-error”法，极其耗时；学习速度慢；容易陷入局部极值；学习可能会不够充分。

4 心得体会

通过本次实验课程，基本上掌握了文本分类的基础算法，通过比较不同模型之间的性能可以发现，深度学习比机器学习在 NLP 领域发挥更突出的作用，同时在实验过程学习了如何进行实验结果分析以及调整参数。