05/26

训练测试：8:2 batch_size:20, epoch: 20, lr=0.01 归一化数据训练 Java, 超过4e5s的进行截断， mse=0.12665 (39.54 hours)， train: 4064 samples, test: 1000 samples Java, 超过4e5s的样本丢弃， mse=0.0902528 (33.38 hours)， train: 3908 samples, test: 960 samples

全部类型, 超过4e5s的进行截断， mse=0.052178096 (25.38 hours)， train: 46220 samples, test: 11540 samples 全部类型, 超过4e5s的样本丢弃， mse=0.0024976 (5.55 hours)， train: 43800 samples, test: 10920 samples

05/28

完善下模型，调下模型参数得到好一些的结果。训练集/测试集比例：9:1。 形成一个实验报告，包含，模型结构图，实验数据介绍，和实验结果数据

数据预处理

1. 实验数据取stackoverflow 2013 前10w条数据，将没有回答者的数据样本剔除后，数据量变为57795条，
2. 剔除post中问题描述中的html标签和其它富文本标签
3. 对数据按照9:1的比例划分训练/测试集，得到训练集 52015条样本。
4. 再次剔除时间跨度大于4e5的样本，得到训练集样本49200条。测试集样本5520条。 【注：按照batch_size=20进行整除】

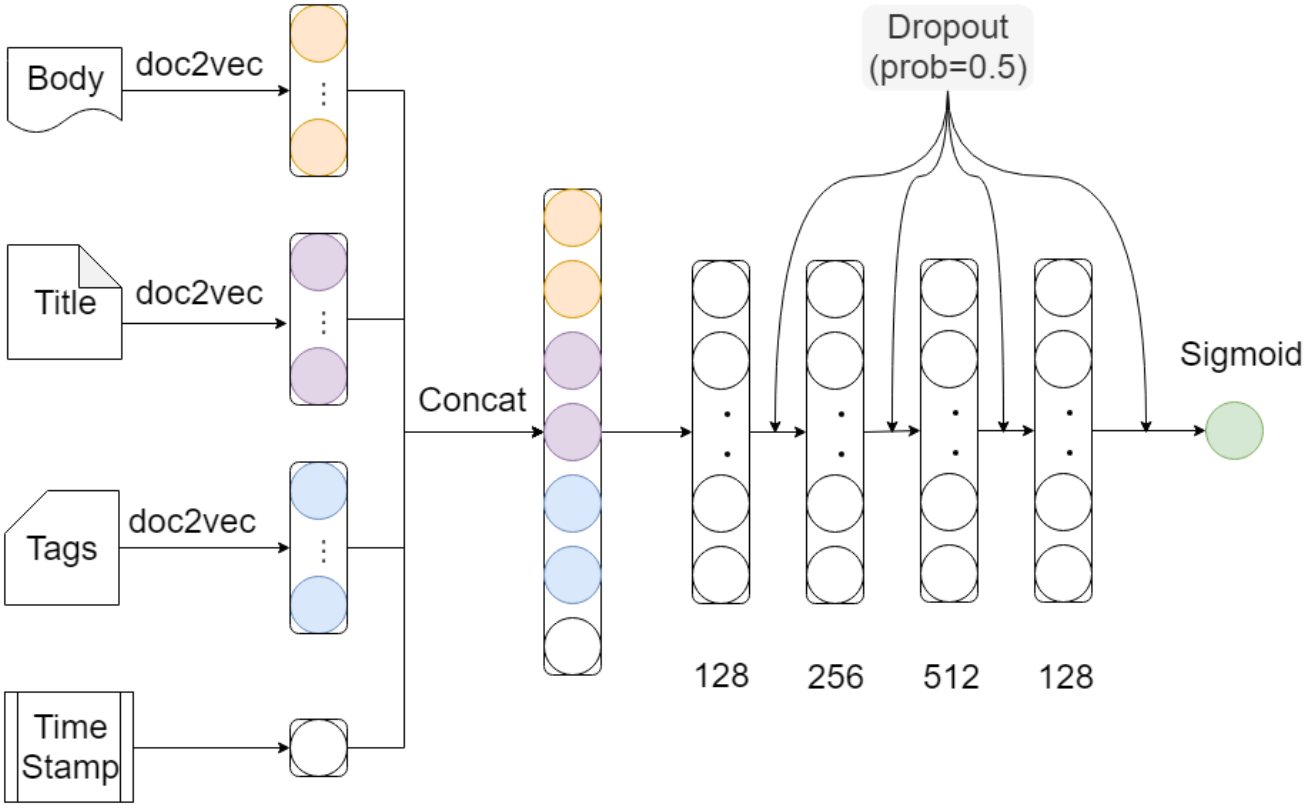
	训练集	**测试集**
剔除无回答者样本	52015	5780
再次剔除时间跨度>4e5 secs样本	49200	5520

5. 使用到的数据特征列：问题从创建到接受第一个回答经过的时间（time）,问题描述（body）,问题标题（title）,问题标签（tags）,问题创建的时间（time-stamp） 对body，title，tags都分别使用doc2vec进行编码，编码向量长度分别设为200，200，200. time和time-stamp都使用归一化的标量值。

注意： doc2vec训练过程使用的是训练和测试的两部分数据，然后使用学习好的模型分别对训练和测试编码向量。 各个不同的特征列训练各自对应的doc2vec编码。

model

全连接网络：将特征列进行拼接，形成601-d的一维向量。



实验结果

batch_size:20, lr=0.01, AdamOptimizer, epoches=60, decay_per_epochs=20

实验1： 考虑到时间戳是个标量，与200维的其它特征列特征表达长度相差太远，不加时间戳特征进行训练

```
# Test data eval
# [get_batches] batch_size: 20, n_batches: 276
# time var:0.02034657087417218
[steps 147600] loss: 0.0029214774258434772
 ------ mean loss: 0.0029320030007511377 ------
```

从测试结果来看，预测时间绝对误差为√(0.0029320030007511377) × 400000 ÷ 3600 = **6.01644** hours。

实验2： 考虑到post创建的时间戳对得到回答的时长有参考价值，加入到输入特征中训练

```
# [get_batches] batch_size: 20, n_batches: 276
# time var:0.02034657087417218
[steps 147600] loss: 0.0029445355758070946
 ------ mean loss: 0.0029222455341368914 ------
```

从测试结果来看，预测时间绝对误差为√(0.0029222455341368914) × 400000 ÷ 3600 = **6.006422** hours。

实验3： 学习率调整，lr由1e-2改为1e-3

```
[steps 147600] loss: 0.002891353564336896
 ------ mean loss: 0.002881178865209222 ------
```

预测时间绝对误差为√(0.002881178865209222) × 400000 ÷ 3600 = **5.964068** hours。

实验4： dropout由0.5改为0.8

```
[steps 147600] loss: 0.0029360875487327576
 ------ mean loss: 0.002906357403844595 ------
```

预测时间绝对误差为√(0.002906357403844595) × 400000 ÷ 3600 = **5.990071** hours。

实验5： 改优化器为SGD（收敛慢但稳，一般能取得最优值，但对参数设置敏感），lr=0.01，每20epoch一次衰减

```
[steps 147600] loss: 0.002890016185119748
 ------ mean loss: 0.002877070801332593 ------
```

预测时间绝对误差为√(0.002877070801332593) × 400000 ÷ 3600 = **5.959815** hours。

实验6：exp_0601001

```
self.keep_prob = 0.8
# feature length
self.body_vec_size = 50
self.title_vec_size = 20
self.tags_vec_size = 5
self.rate_vec_size = 1
```

lr=0.01, AdamOptimizer

```
with tf.variable_scope("predictor", initializer=self.initializer):
    # note: self.input_data shape is (batch, time_steps, feature_len), in which, feature dim is
`b,ir,or`
    print("[build_embedding_layer] input_data shape {}".format(self.input_data.get_shape()))
    input_embedding = tf.layers.dense(inputs=self.input_data, units=128,
activation=cfg.hidden_act)

    input_dropout = tf.layers.dropout(input_embedding, rate=self.dropout_prob())
    layer1 = tf.layers.dense(input_dropout, 256, activation=cfg.hidden_act)
    layer1_dropout = tf.layers.dropout(layer1, rate=self.dropout_prob())
    layer2 = tf.layers.dense(layer1_dropout, 512, activation=cfg.hidden_act)
    layer2_dropout = tf.layers.dropout(layer2, rate=self.dropout_prob())
    layer3 = tf.layers.dense(layer2_dropout, 128, activation=cfg.hidden_act)
    layer3_dropout = tf.layers.dropout(layer3, rate=self.dropout_prob())

    # softmax layer return prob distribution
    self.prediction = tf.layers.dense(layer3_dropout, 1, activation=cfg.output_act,
name="predictions")
```

```
[steps 147600] loss: 0.002984224585816264
 ------ mean loss: 0.002973471535369754 ------
```

实验7： 减少模型全连接层数，降低神经元个数

```
with tf.variable_scope("predictor", initializer=self.initializer):
    input_embedding = tf.layers.dense(inputs=self.input_data, units=100,
activation=cfg.hidden_act)

    input_dropout = tf.layers.dropout(input_embedding, rate=self.dropout_prob())
    layer1 = tf.layers.dense(input_dropout, 200, activation=cfg.hidden_act)
    layer1_dropout = tf.layers.dropout(layer1, rate=self.dropout_prob())
    layer2 = tf.layers.dense(layer1_dropout, 100, activation=cfg.hidden_act)
    layer2_dropout = tf.layers.dropout(layer2, rate=self.dropout_prob())

    # softmax layer return prob distribution
    self.prediction = tf.layers.dense(layer2_dropout, 1, activation=cfg.output_act,
name="predictions")
```

lr=0.01, batch_size=20, Adam, dropout_prob=0.8

```
[steps 147600] loss: 0.002658157143741846
 ------ mean loss: 0.0025460838805884123 ------
```

绝对误差时间 √(0.0025460838805884123) × 400000 ÷ 3600 = 5.606526 hours

==========实验7附加========== 对该模型神经元个数改为 256 -> 128 -> 64 后，

```
[steps 147600] loss: 0.0026180557906627655
 ------ mean loss: 0.0025569377467036247 ------
```

测试绝对误差时间 √(0.0025569377467036247) × 400000 ÷ 3600 = 5.618464 hours

在实验7基础上，进行组合实验，目的： 找出重要的特征。 实验7.1： 去除时间戳特征， 即只使用body， title， tags特征

```
[steps 147600] loss: 0.0025346584152430296
 ------ mean loss: 0.0025752766523510218 ------
```

实验7.2： 去除tags特征， 即只使用body， title， time-rate特征

```
[steps 147600] loss: 0.0025575035251677036
 ------ mean loss: 0.0025597945787012577 ------
```

实验7.3： 去除title特征， 即只使用body， tags， time-rate特征

```
[steps 147600] loss: 0.0025929848197847605
 ------ mean loss: 0.0025811747182160616 ------
```

实验7.4： 去除body特征， 即只使用title， tags， time-rate特征

```
[steps 147600] loss: 0.0033198464661836624
 ------ mean loss: 0.0032802806235849857 ------
```

结论：1.减小模型〔使用轻量级模型〕能提高预测结果 2.doc2vec编码长短对预测结果影响甚微 3.删除body特征列会扩大预测损失，而删除其它特征列扩大程度较小...即body属于最重要一类特征，其它特征可选其一