

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】

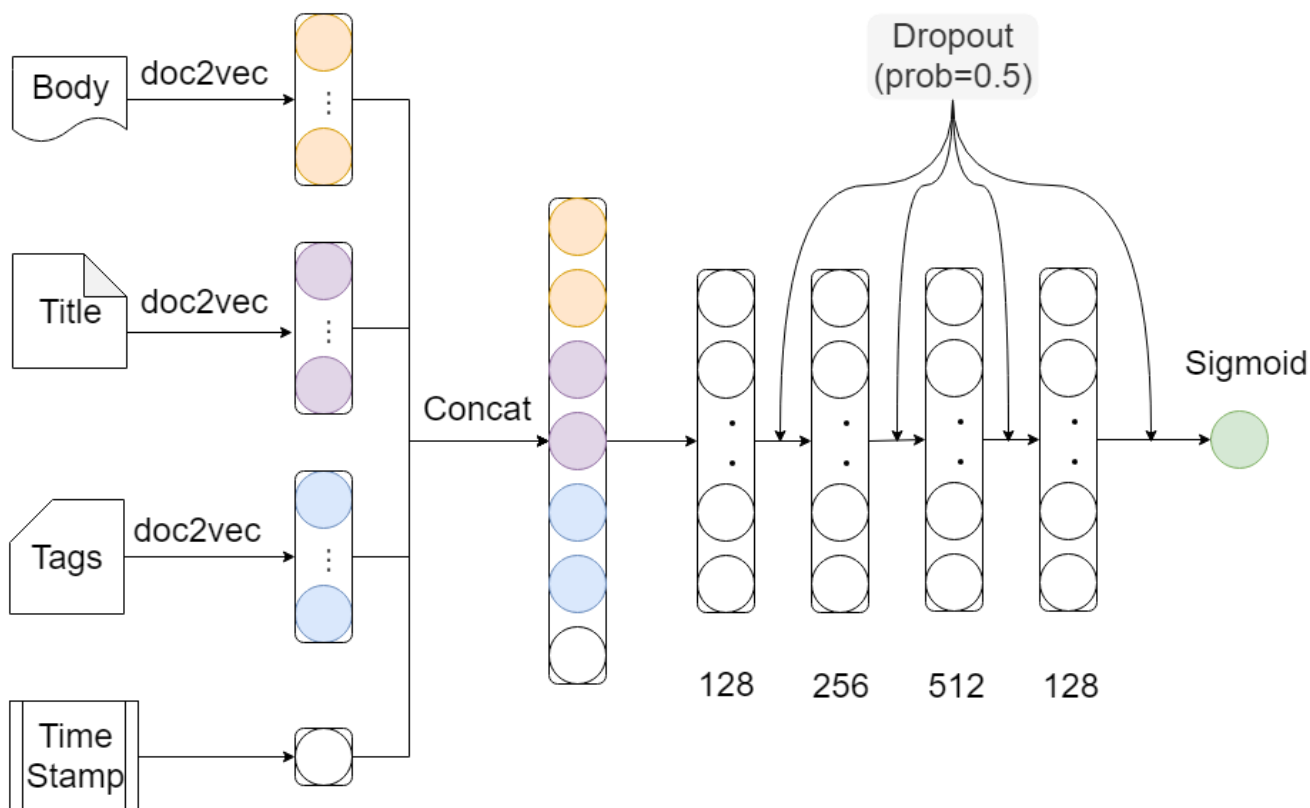
| | 训练集 | 测试集 |
|---------------------|-------|------|
| 剔除无回答者样本 | 52015 | 5780 |
| 再次剔除时间跨度>4e5 secs样本 | 49200 | 5520 |

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 $\sqrt{(0.0029320030007511377) \times 400000 \div 3600} = \mathbf{6.01644}$ 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 $\sqrt{(0.0029222455341368914) \times 400000 \div 3600} = \mathbf{6.006422}$ 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 $\sqrt{(0.002881178865209222) \times 400000 \div 3600} = \mathbf{5.964068}$ hours。

实验3: 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 $\sqrt{(0.002906357403844595) \times 400000 \div 3600} = \mathbf{5.990071}$ hours。

实验4: 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5: 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 $\sqrt{(0.0025460838805884123) \times 400000 \div 3600} = 5.606526$ hours

=====实验5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 $\sqrt{(0.0025569377467036247) \times 400000 \div 3600} = 5.618464$ hours

在实验5基础上, 进行组合实验, 目的: 找出重要的特征。实验5.1: 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5.2: 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5.3: 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5.4: 去除body特征, 即只使用title, tags, time-rate特征

```

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

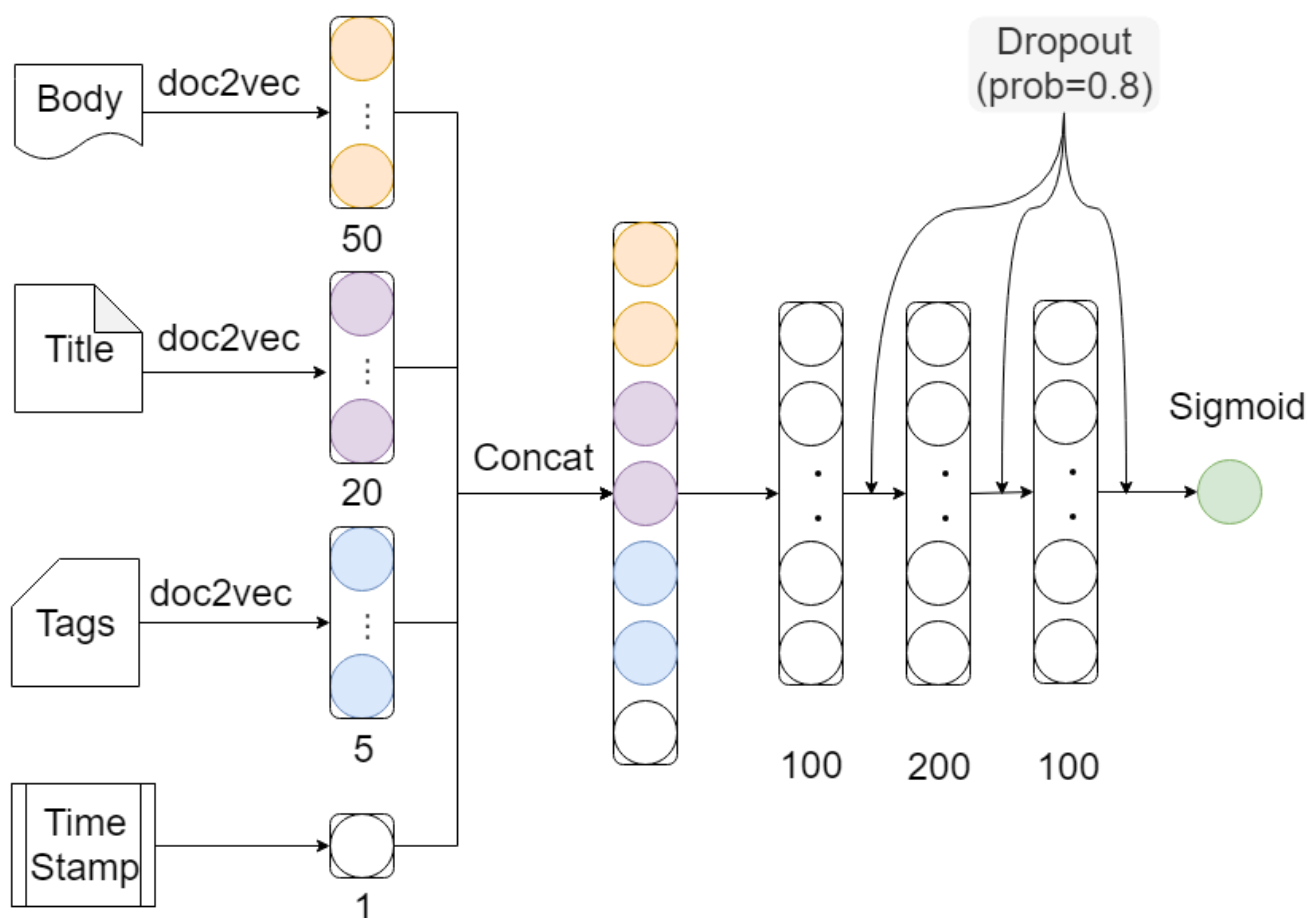
```

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

实验5汇总表格

| Body | Title | Tags | Time-rate | 预测绝对误差(hours) |
|------|-------|------|-----------|---------------|
| | | | × | 5.638576 |
| | | × | | 5.621601 |
| | × | | | 5.645029 |
| × | | | | 6.363748 |
| | | | | 5.606526 |

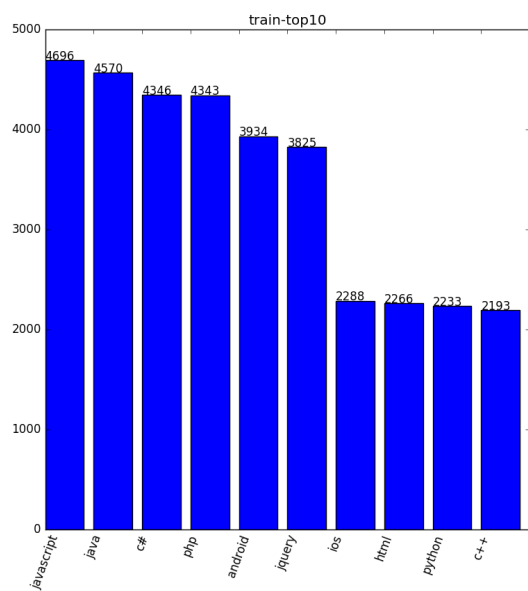
模型（下方数字表示向量长度）



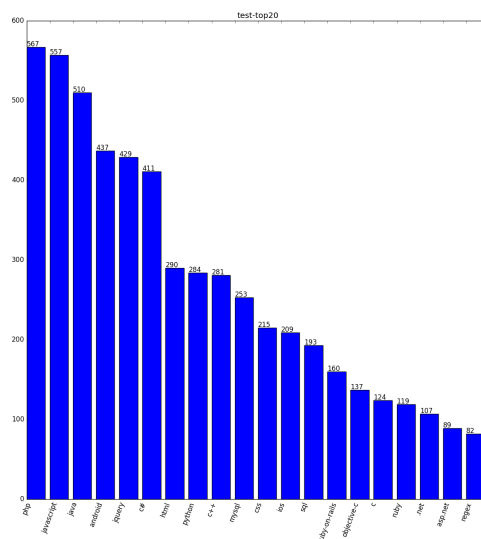
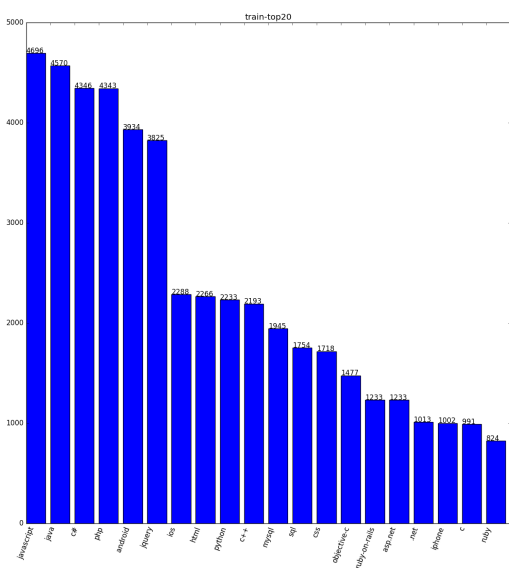
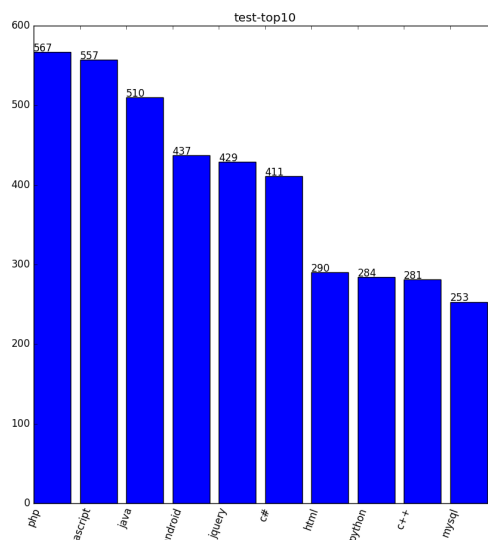
0604 按照Tags归类的数据分布

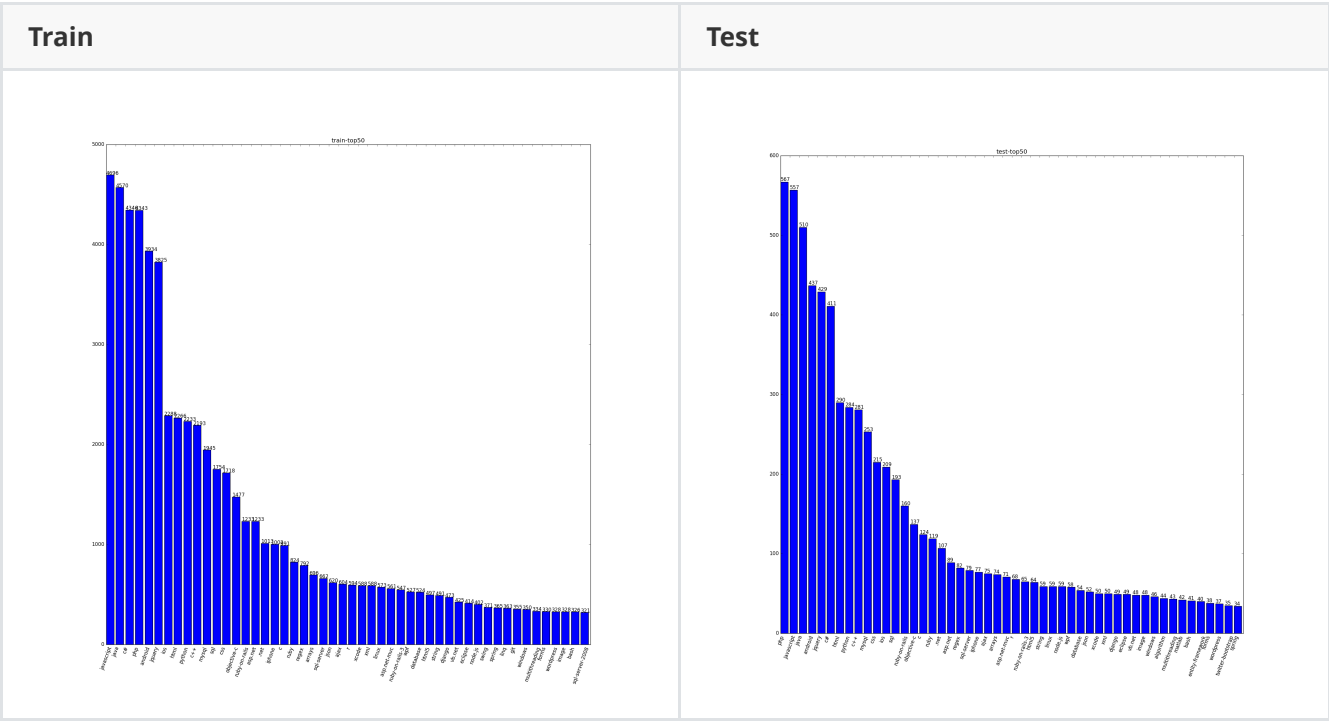
按照Tags归类的数据分布，从上到下依次为top10，top20，top50

Train



Test





实验6:

加入星期特征: [steps 147600] loss: 0.002624884946271777 ----- mean loss: 0.0025482838973402977 -----

删除tags [steps 147600] loss: 0.002507805125787854 ----- mean loss: 0.002554275095462799 -----

删除title

[steps 147600] loss: 0.002513353480026126 ----- mean loss: 0.002534728730097413 -----

删除body

[steps 147600] loss: 0.0032521190587431192 ----- mean loss: 0.003240973223000765 -----

实验6汇总表格

| Body | Title | Tags | Time-rate+weekday | 预测绝对误差(hours) |
|------|-------|------|-------------------|---------------|
| | | × | | 5.615537 |
| | × | | | 5.594010 |
| × | | | | 6.325505 |
| | | | | 5.608947 |

实验7： 其他sklearn中的回归算法（使用body， Title， Tags， Timerate+weekday特征）

注： 因为不是使用mini-batch方式拟合训练，因此数据量会有微小的差距（get_batch方式对于不满足一个batch大小的最后一批数据使用抛弃策略）

训练集：

time var:0.016922068533101995

X.shape=(49214, 77); y.shape= (49214,)

测试集：

time var:0.020330074736748433 X.shape=(5525, 77); y.shape= (5525,)

| 回归模型 | 预测绝对误差(hours) |
|-----------------------|--------------------------------------|
| LinearRegression | 15.533570 (mse=0.0195446345324386) |
| KNeighborsRegressor | 16.545609 (mse=0.02217433114389324) |
| RandomForestRegressor | 16.673747 (mse=0.022519120022870463) |
| SVR | 16.539896 (mse=0.02215902253231699) |
| MLPRegressor | 15.923066 (mse=0.02053706645361) |