In [1]:
```
%load_ext autoreload
%autoreload 2
```

In [2]:
```
from MultiStepLSTM import WirelineLog, MultiStepLTSM
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, scale, StandardScaler
from keras.models import load_model
```

```
Using TensorFlow backend.
```

In [ ]:
```
data = WirelineLog()
data2 = WirelineLog()
data3 = WirelineLog()
#data.read("/home/duys/Downloads/ccl_data/stage2b-pass4.las")
data.read("/home/duys/Downloads/anadarko.las")
data2.read("/home/duys/Downloads/ccl_data/stage2b-pass4.las")
data3.read("/home/duys/Downloads/anadarko2.las")

d = data.df['CCL'].dropna().values
d2 = data2.df['CCL'].dropna().values
d3 = data3.df['CCL'].dropna().values

scaler = StandardScaler()

d = scale(d)
d2 = scale(d2)
d3=scale(d3)

ccl_data =  {'train': d[3000:5000], 'train2' : d2, 'train3' : d3}
# plt.figure(figsize=(100,10))
# plt.plot(ccl_data['train'][4500:5000])
# plt.plot(ccl_data['train2'][4500:5000])
# plt.plot(ccl_data['train3'][:500])
model2 = MultiStepLTSM(data_dict=ccl_data, batch_size=25, look_back=100, epoch
s=15, hidden_n=100, look_ahead=1)
model2.build_model(iterations=0)
model2.preprocess_data(scale_data=False)
model2.train_model()
model2.predict_on(model2.data['train']['x'], name='train', inverse_transform=Fa
lse)
model2.plot_error('train', 'train', i1=200, i2=500)
```

Steps to Training

0) Split data into 4 subsets

    a) Train, Test, Validation1, Validation2

1) Train Model on non-ccl kicks (data inbetween collars) scaled to mean=0, variance=1

2) Calcualte anomalies (p-value) on actual ccl data

    a) collars will show up as anomalies,but severity of anoamly is what matters

Split data into 4 subsets

In [3]:
```
from sklearn.model_selection import train_test_split

ccl_log = WirelineLog()
ccl_log.read("/home/duys/Downloads/anadarko.las")
ccl = ccl_log.df['CCL'].dropna().values
```

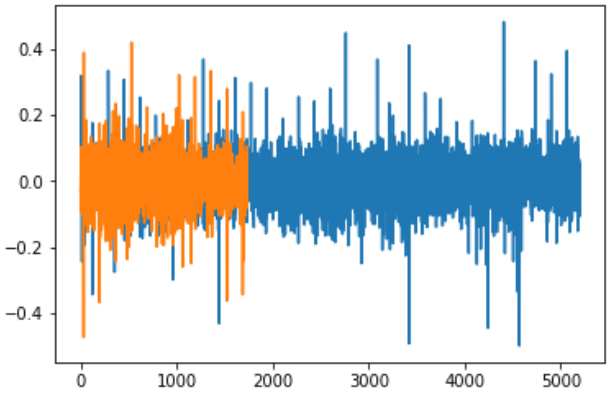
In [4]:
```
non_anomaly_dset = []

for i in range(len(ccl)/2):
    if ccl[i] > 0.5 or ccl[i] < -0.5:
        continue
    non_anomaly_dset.append(ccl[i])
non_anomaly_dset = np.array(non_anomaly_dset)
non_anomaly_dset.shape
```

Out[4]:
```
(6931,)
```

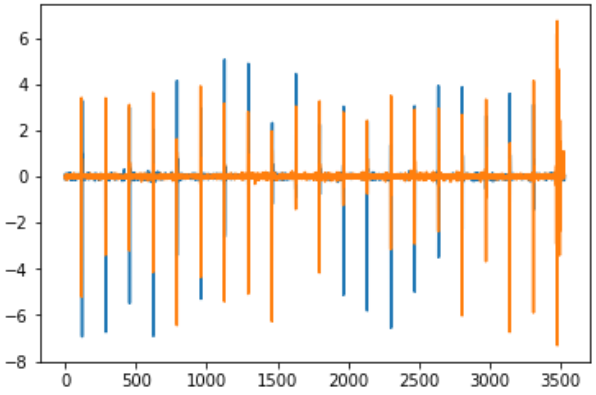
In [5]:
```
train, validation1 = train_test_split(non_anomaly_dset, shuffle=False, test_siz
e=0.25)
plt.plot(train)
plt.plot(validation1)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x7f34b1cc2ad0>]
```



In [6]:
```
other_half = ccl[int(len(ccl)/2):]
test, validation2 = train_test_split(other_half, test_size=0.5, shuffle=False)
plt.plot(test)
plt.plot(validation2)
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x7f34b1be3e90>]
```



In [7]:
```
ccl_data = {
    "train" : train, #non-anomalous data
    "validation_1" : validation1, # also non-anomalous data
    "test": test, # anomalous data mix
    "validation_2": validation2
}
```

Build Model with the split data

In [8]:
```
model2 = MultiStepLTSM(data_dict=ccl_data, batch_size=250, look_back=100, epoch
s=5, hidden_n=120, look_ahead=1)
model2.build_model(iterations=0)
model2.preprocess_data(scale_data=False)
```

```
WARNING: Logging before flag parsing goes to stderr.
W0705 21:52:42.603987 139866887808832 deprecation_wrapper.py:119] From /home/du
ys/Documents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site
-packages/keras/backend/tensorflow_backend.py:47: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0705 21:52:42.611819 139866887808832 deprecation_wrapper.py:119] From /home/du
ys/Documents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site
-packages/keras/backend/tensorflow_backend.py:351: The name tf.placeholder is d
eprecated. Please use tf.compat.v1.placeholder instead.

W0705 21:52:42.651843 139866887808832 deprecation.py:506] From /home/duys/Docum
ents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site-package
s/keras/backend/tensorflow_backend.py:521: calling __init__ (from tensorflow.py
thon.ops.init_ops) with dtype is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the
constructor
W0705 21:52:42.726841 139866887808832 deprecation_wrapper.py:119] From /home/du
ys/Documents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site
-packages/keras/backend/tensorflow_backend.py:3176: The name tf.random_uniform
is deprecated. Please use tf.random.uniform instead.

W0705 21:52:42.873050 139866887808832 deprecation_wrapper.py:119] From /home/du
ys/Documents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site
-packages/keras/backend/tensorflow_backend.py:141: The name tf.get_default_sess
ion is deprecated. Please use tf.compat.v1.get_default_session instead.

W0705 21:52:42.874638 139866887808832 deprecation_wrapper.py:119] From /home/du
ys/Documents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site
-packages/keras/backend/tensorflow_backend.py:146: The name tf.ConfigProto is d
eprecated. Please use tf.compat.v1.ConfigProto instead.

W0705 21:52:54.367542 139866887808832 deprecation.py:506] From /home/duys/Docum
ents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site-package
s/keras/backend/tensorflow_backend.py:2711: calling dropout (from tensorflow.py
thon.ops.nn_ops) with keep_prob is deprecated and will be removed in a future v
ersion.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - kee
p_prob`.
W0705 21:52:54.606848 139866887808832 deprecation_wrapper.py:119] From /home/du
ys/Documents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site
-packages/keras/optimizers.py:675: The name tf.train.Optimizer is deprecated. P
lease use tf.compat.v1.train.Optimizer instead.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (250, 100, 120)           58560
_________________________________________________________________
dropout_1 (Dropout)          (250, 100, 120)           0
_________________________________________________________________
lstm_2 (LSTM)                (250, 120)                115680
_________________________________________________________________
dense_1 (Dense)              (250, 1)                  121
=================================================================
Total params: 174,361
Trainable params: 174,361
Non-trainable params: 0
_________________________________________________________________
```

In [9]:
```
model2.train_model()
```

```
Epoch 1 / 5

W0705 21:54:58.980762 139866887808832 deprecation.py:323] From /home/duys/Docum
ents/machine_learning/lstm_anomaly_thesis/lstm/local/lib/python2.7/site-package
s/tensorflow/python/ops/math_grad.py:1250: where (from tensorflow.python.ops.ar
ray_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where

Train on 5000 samples, validate on 1500 samples
Epoch 1/1
5000/5000 [==============================] - 4s - loss: 0.0045 - mean_absolute_
error: 0.0502 - val_loss: 0.0051 - val_mean_absolute_error: 0.0532
Epoch 2 / 5
Train on 5000 samples, validate on 1500 samples
Epoch 1/1
5000/5000 [==============================] - 2s - loss: 0.0043 - mean_absolute_
error: 0.0488 - val_loss: 0.0048 - val_mean_absolute_error: 0.0513
Epoch 3 / 5
Train on 5000 samples, validate on 1500 samples
Epoch 1/1
5000/5000 [==============================] - 2s - loss: 0.0042 - mean_absolute_
error: 0.0481 - val_loss: 0.0046 - val_mean_absolute_error: 0.0507
Epoch 4 / 5
Train on 5000 samples, validate on 1500 samples
Epoch 1/1
5000/5000 [==============================] - 2s - loss: 0.0043 - mean_absolute_
error: 0.0481 - val_loss: 0.0047 - val_mean_absolute_error: 0.0508
Epoch 5 / 5
Train on 5000 samples, validate on 1500 samples
Epoch 1/1
5000/5000 [==============================] - 2s - loss: 0.0043 - mean_absolute_
error: 0.0481 - val_loss: 0.0047 - val_mean_absolute_error: 0.0508
```

In [10]:
```
x_test, y_test = model2.data['test']['x'], model2.data['test']['y']
valid2_x, valid2_y = model2.data['validation_2']['x'], model2.data['validation_
2']['y']
batch_size = model2.batch_size

test_loss = model2.evaluate(x_test, y_test[:,0], batch_size=batch_size)
validation2_loss = model2.evaluate(valid2_x, valid2_y[:,0], batch_size=batch_si
ze)
print(test_loss)
print(validation2_loss)
```

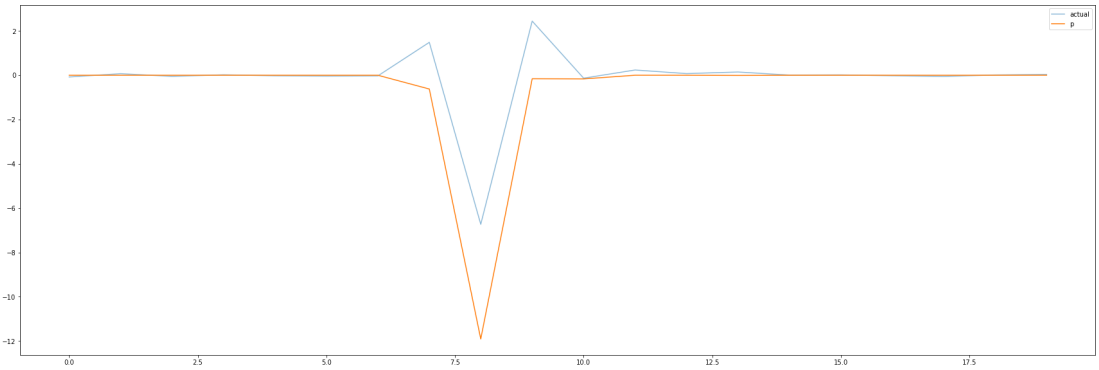
```
3250/3250 [==============================] - 0s
3250/3250 [==============================] - 0s
[0.1734924022681438, 0.09410931915044785]
[0.16158901384243599, 0.09327439161447379]
```

In [11]:
```
model2.predict_on(model2.data['test']['x'], name='test', inverse_transform=Fals
e)
```

In [14]:
```
from_n = 180
to_n = 200

y_pred = model2.predictions['test'][from_n:to_n]
y_actual = model2.data['test']['y'][from_n:to_n,0]
plt.figure(figsize=(30,10))
# plt.ylim(-1,4)
plt.plot(y_actual, label='actual', alpha=0.5)
#plt.plot(y_pred, label='pred')
#plt.plot(abs(y_pred-y_actual))
p_values = model2.calc_pvalues("test", "test")/10.0
plt.plot(p_values[from_n:to_n], label='p')
plt.legend()
plt.show()
```

```
(3250, 1) (3250,) 1
```



In [15]:
```
model2.save("ccl_test.keras-model")
```

Test trained model on new ccl data from another log

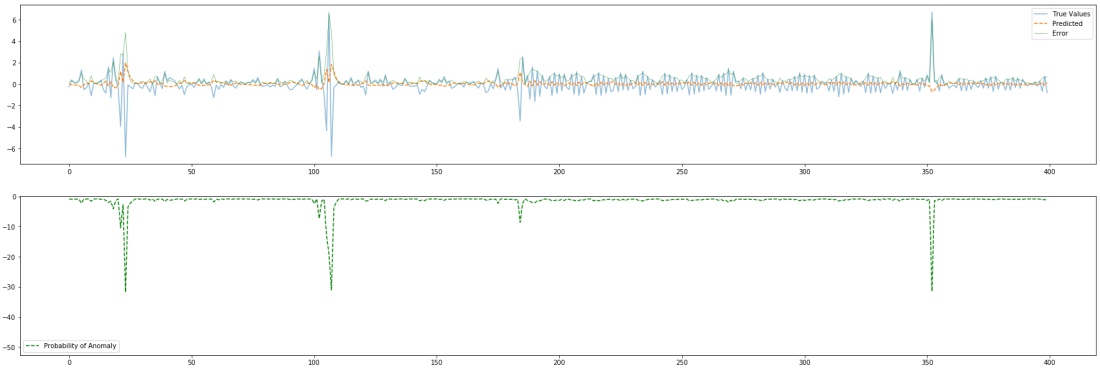
In [33]:
```
# model2.predict_on(model2.data['test']['x'], name='test', inverse_transform=Fa
lse)
#model2 = load_model('ccl_test.keras-model', custom_objects={'MultiStepLTSM': M
ultiStepLTSM})
anadarko_log = WirelineLog()
anadarko_log.read("/home/duys/Downloads/anadarko2.las")
new_ccl = scale(anadarko_log.df['CCL'].dropna().values)
```

In [34]:
```
model2.insert_data({'new_ccl' : new_ccl})
```

In [35]:
```
model2.predict_on(model2.data['new_ccl']['x'], name='external_data', inverse_tr
ansform=False)
```

In [36]:
```
model2.plot_error(true_name="new_ccl", pred_name="external_data", i1=100, i
2=500)
```

```
(3750, 1) (3750,) 1
```



Test trained model on a second different ccl data from different log

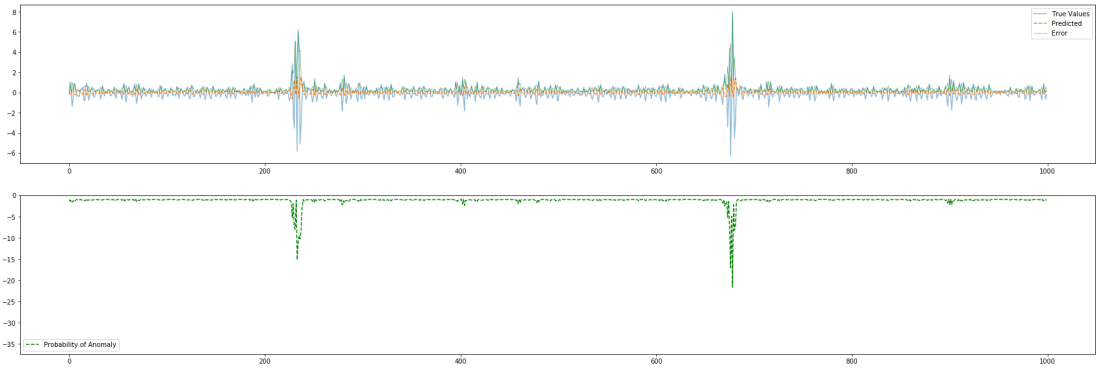
In [26]:
```
stage_log = WirelineLog()
stage_log.read("/home/duys/Downloads/ccl_data/stage2b-pass4.las")
stage_ccl = scale(stage_log.df['CCL'].dropna().values)
```

In [27]:
```
model2.insert_data({'stage_ccl' : stage_ccl})
```

In [28]:
```
model2.predict_on(model2.data['stage_ccl']['x'], name='external_data2', inverse
_transform=False)
```

In [31]:
```
model2.plot_error(true_name='stage_ccl',pred_name='external_data2', i1=4500, i
2=5500)
```

```
(81250, 1) (81250,) 1
```



In [ ]: