In [1]:

```
import ktrain
from ktrain import text
import pandas as pd
import random
import numpy as np
import math
```

In [2]:

```
csv_file = '../../data/merged_ktrain_four_six_months.csv'
data = pd.read_csv(csv_file).values
print(len(data))
```

```
30500
```

In [3]:

```
epochs = 15
learning_rate = 5e-5
batch_size = 64
max_length = 10
max_words = 25000
```

In [4]:

```
def split_test_data(data, split=0.1, random_seed=42):
    np.random.seed(random_seed)
    np.random.shuffle(data)
    split_item = math.floor(split * len(data))
    print('split at: ', split_item)
    x_test, y_test = data[:split_item, 0], data[:split_item, 1:]
    x_train, y_train = data[split_item:, 0], data[split_item:, 1:]
    return x_train, y_train, x_test, y_test
```

In [5]:

```
x_train, y_train, x_val, y_val = split_test_data(data, split=0.15, random_seed=4
242)
print(len(x_train), len(y_train), len(x_val), len(y_val))
```

```
split at:  4575
25925 25925 4575 4575
```

In [6]:

```
from sklearn.utils import class_weight
def generate_balanced_weights(y_train):
    y_labels = [y.argmax() for y in y_train]
    class_weights = class_weight.compute_class_weight('balanced', np.unique(y_la
bels), y_labels)
    weight_dict = {}
    for key in range(len(class_weights)):
        weight_dict[key] = class_weights[key]
    return weight_dict

class_weight_dict = generate_balanced_weights(y_train)
print(class_weight_dict)
```

```
{0: 1.9312425506555424, 1: 0.6106897201545275, 2: 0.751797935274330
1, 3: 1.9434032983508245}
```

In [7]:

```
MODEL ='distilbert-base-multilingual-cased'
MODEL_bert = 'bert-base-german-cased'
transformer = text.Transformer(MODEL_bert, maxlen=max_length, class_names=['leas
t','less','more','most'])
train_data = transformer.preprocess_train(x_train, y_train)
val_data = transformer.preprocess_test(x_val, y_val)
```

```
preprocessing train...
language: de
train sequence lengths:
        mean : 6
        95percentile : 9
        99percentile : 11


Is Multi-Label? False
preprocessing test...
language: de
test sequence lengths:
        mean : 6
        95percentile : 9
        99percentile : 11
```

In [8]:

```
model = transformer.get_classifier()
```

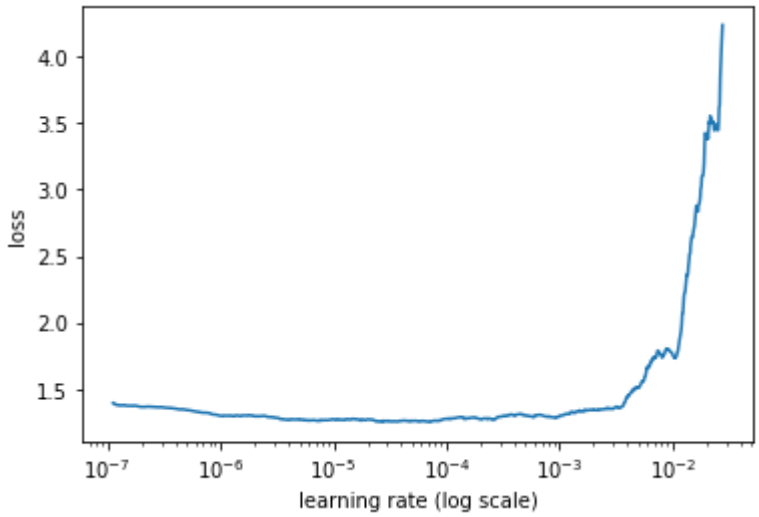
In [9]:

```
learner = ktrain.get_learner(model, train_data=train_data, val_data=val_data, ba
tch_size=batch_size)
```

In [17]:

```
learner.lr_find(show_plot=True, max_epochs=2)
```

```
simulating training for different learning rates... this may take a
few moments...
Train for 810 steps
Epoch 1/2
810/810 [==============================] - 71s 88ms/step - loss: 1.2
959 - accuracy: 0.3791
Epoch 2/2
298/810 [==========>...................] - ETA: 42s - loss: 2.6083 -
accuracy: 0.3364

done.
Visually inspect loss plot and select learning rate associated with
falling loss
```



In [10]:

```
learner.fit_onecycle(5e-5, epochs=epochs, class_weight=class_weight_dict)
```

```
begin training using onecycle policy with max lr of 5e-05...
Train for 406 steps, validate for 143 steps
Epoch 1/15
406/406 [==============================] - 64s 158ms/step - loss: 1.
3876 - accuracy: 0.2551 - val_loss: 1.3681 - val_accuracy: 0.2450
Epoch 2/15
406/406 [==============================] - 50s 122ms/step - loss: 1.
3620 - accuracy: 0.2942 - val_loss: 1.3696 - val_accuracy: 0.2896
Epoch 3/15
406/406 [==============================] - 50s 123ms/step - loss: 1.
3010 - accuracy: 0.3490 - val_loss: 1.4191 - val_accuracy: 0.3322
Epoch 4/15
406/406 [==============================] - 50s 123ms/step - loss: 1.
1614 - accuracy: 0.4299 - val_loss: 1.5284 - val_accuracy: 0.2957
Epoch 5/15
406/406 [==============================] - 49s 122ms/step - loss: 0.
9466 - accuracy: 0.5300 - val_loss: 1.7035 - val_accuracy: 0.3193
Epoch 6/15
406/406 [==============================] - 50s 124ms/step - loss: 0.
7342 - accuracy: 0.6274 - val_loss: 2.0130 - val_accuracy: 0.3128
Epoch 7/15
406/406 [==============================] - 50s 123ms/step - loss: 0.
5742 - accuracy: 0.7111 - val_loss: 2.3429 - val_accuracy: 0.3650
Epoch 8/15
406/406 [==============================] - 48s 119ms/step - loss: 0.
4688 - accuracy: 0.7693 - val_loss: 2.6045 - val_accuracy: 0.3480
Epoch 9/15
406/406 [==============================] - 50s 124ms/step - loss: 0.
3446 - accuracy: 0.8342 - val_loss: 2.7255 - val_accuracy: 0.3233
Epoch 10/15
406/406 [==============================] - 49s 121ms/step - loss: 0.
2420 - accuracy: 0.8874 - val_loss: 3.1780 - val_accuracy: 0.3580
Epoch 11/15
406/406 [==============================] - 48s 119ms/step - loss: 0.
1754 - accuracy: 0.9211 - val_loss: 3.5910 - val_accuracy: 0.3692
Epoch 12/15
406/406 [==============================] - 49s 121ms/step - loss: 0.
1309 - accuracy: 0.9409 - val_loss: 3.8753 - val_accuracy: 0.3753
Epoch 13/15
406/406 [==============================] - 49s 121ms/step - loss: 0.
0950 - accuracy: 0.9562 - val_loss: 4.0577 - val_accuracy: 0.3738
Epoch 14/15
406/406 [==============================] - 50s 123ms/step - loss: 0.
0758 - accuracy: 0.9669 - val_loss: 4.1441 - val_accuracy: 0.3773
Epoch 15/15
406/406 [==============================] - 47s 115ms/step - loss: 0.
0634 - accuracy: 0.9719 - val_loss: 4.3440 - val_accuracy: 0.3808
```

Out[10]:

```
<tensorflow.python.keras.callbacks.History at 0x7f8a50312f60>
```

In [11]:

```
predictor = ktrain.get_predictor(learner.model, preproc=transformer)
```

In [12]:

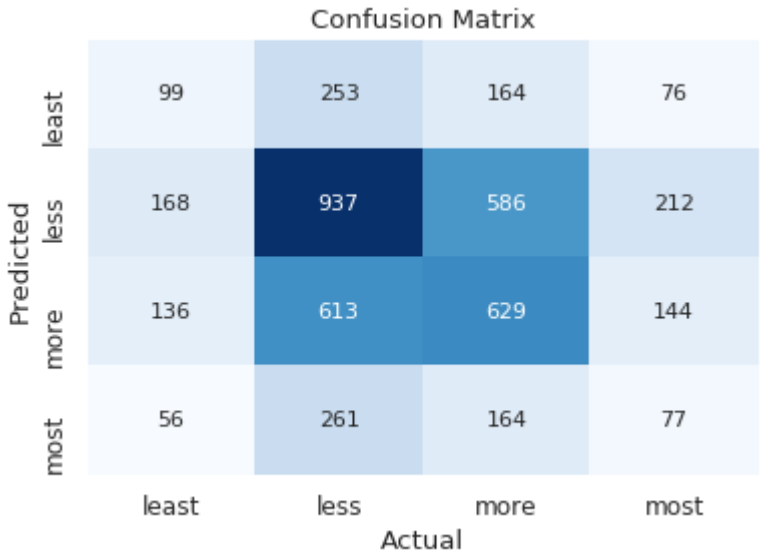
```
confusion = learner.evaluate()
```

```
              precision    recall  f1-score   support

           0       0.22      0.17      0.19       592
           1       0.45      0.49      0.47      1903
           2       0.41      0.41      0.41      1522
           3       0.15      0.14      0.14       558

    accuracy                           0.38      4575
   macro avg       0.31      0.30      0.30      4575
weighted avg       0.37      0.38      0.38      4575
```

In [13]:

```
# print confusion matrix
import matplotlib.pyplot as plt
import seaborn as sn
labels = ['least','less','more','most']
cm_df = pd.DataFrame(confusion, labels, labels)
sn.set(font_scale=1.1, font='Arial')
ax = sn.heatmap(cm_df, cmap="Blues", annot=True, annot_kws={"size": 11}, cbar=Fa
lse, fmt='g')
ax.set_xlabel("Actual")
ax.set_ylabel("Predicted")
ax.set_title("Confusion Matrix")
plt.show()
```

```
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sa
ns.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sa
ns.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sa
ns.
```

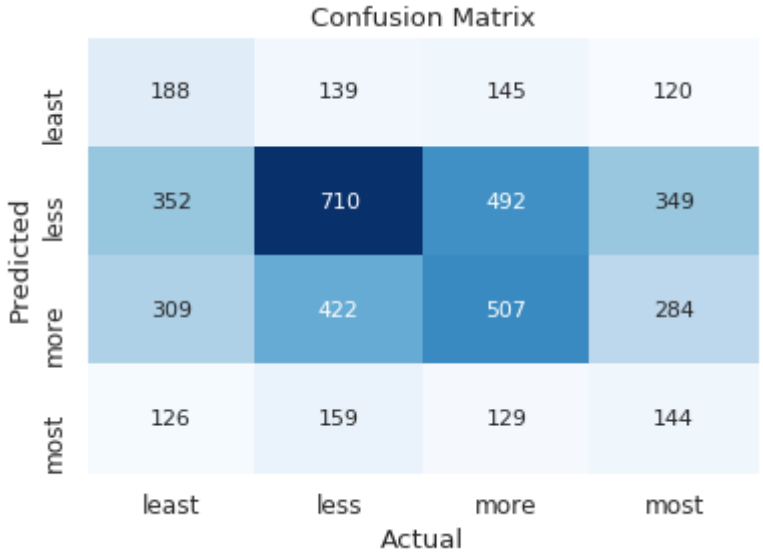


In [26]:

```
confusion = learner.evaluate()
```

```
              precision    recall  f1-score   support

           0       0.19      0.32      0.24       592
           1       0.50      0.37      0.43      1903
           2       0.40      0.33      0.36      1522
           3       0.16      0.26      0.20       558

    accuracy                           0.34      4575
   macro avg       0.31      0.32      0.31      4575
weighted avg       0.38      0.34      0.35      4575
```

In [28]:



In [ ]:

In [ ]: