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

```
30500
```

In [3]:

```
epochs = 4
learning_rate = 5e-5
batch_size = 32
max_length = 21
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=0.05, random_seed=4
242)
print(len(x_train), len(y_train), len(x_val), len(y_val))
```

```
split at:  1525
28975 28975 1525 1525
```

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: 1.9326974386339382, 1: 0.6088208102202051, 2: 0.751270483302219
5, 3: 1.9646731760238676}
```

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()
```

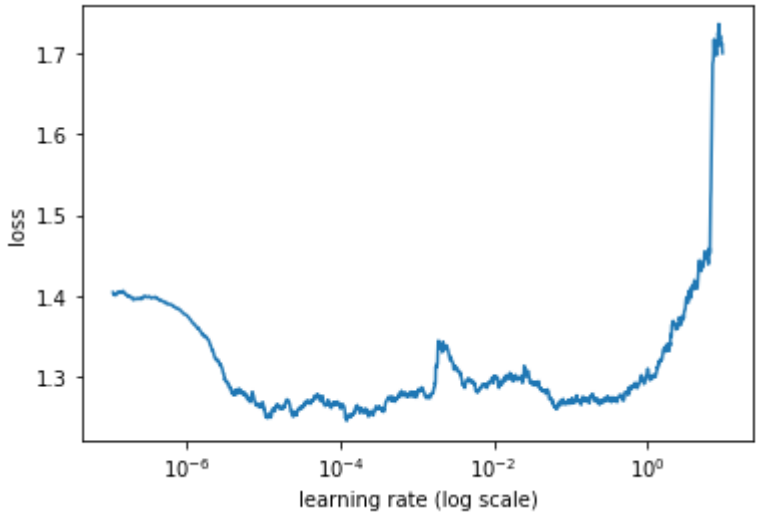
In [9]:

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

In [10]:

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

```
simulating training for different learning rates... this may take a
few moments...
Train for 905 steps
Epoch 1/2
905/905 [==============================] - 88s 97ms/step - loss: 1.3
021 - accuracy: 0.3753
Epoch 2/2
905/905 [==============================] - 85s 93ms/step - loss: 1.3
460 - accuracy: 0.3904


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



In [11]:

```
learner.fit_onecycle(4e-5, epochs=epochs, class_weight=class_weight_dict)
```

```
begin training using onecycle policy with max lr of 2e-05...
Train for 906 steps, validate for 48 steps
Epoch 1/4
906/906 [==============================] - 95s 105ms/step - loss: 1.
3840 - accuracy: 0.2579 - val_loss: 1.3952 - val_accuracy: 0.1836
Epoch 2/4
906/906 [==============================] - 86s 95ms/step - loss: 1.3
640 - accuracy: 0.2717 - val_loss: 1.3977 - val_accuracy: 0.3030
Epoch 3/4
906/906 [==============================] - 87s 96ms/step - loss: 1.2
978 - accuracy: 0.3354 - val_loss: 1.4002 - val_accuracy: 0.3121
Epoch 4/4
906/906 [==============================] - 88s 97ms/step - loss: 1.1
208 - accuracy: 0.4448 - val_loss: 1.5193 - val_accuracy: 0.3167
```

Out[11]:

```
<tensorflow.python.keras.callbacks.History at 0x7f533c466f60>
```

In [13]:

```
learner.view_top_losses(n=10, preproc=transformer)
```

```
----------
id:382 | loss:1.23 | true:equal | pred:more)

----------
id:454 | loss:1.23 | true:equal | pred:more)

----------
id:109 | loss:1.22 | true:equal | pred:more)

----------
id:608 | loss:1.22 | true:equal | pred:more)

----------
id:68 | loss:1.22 | true:equal | pred:more)

----------
id:720 | loss:1.22 | true:equal | pred:more)

----------
id:385 | loss:1.22 | true:equal | pred:more)

----------
id:520 | loss:1.22 | true:equal | pred:more)

----------
id:512 | loss:1.22 | true:equal | pred:more)

----------
id:761 | loss:1.21 | true:equal | pred:more)
```

In [12]:

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

In [15]:

```
predictor.explain(x_train[741])
```

```
/home/sebastian/.local/lib/python3.8/site-packages/ktrain/text/predi
ctor.py:112: UserWarning: ktrain requires a forked version of eli5 t
o support tf.keras. Install with: pip3 install git+https://github.co
m/amaiya/eli5@tfkeras_0_10_1
  warnings.warn(msg)
```

In [13]:

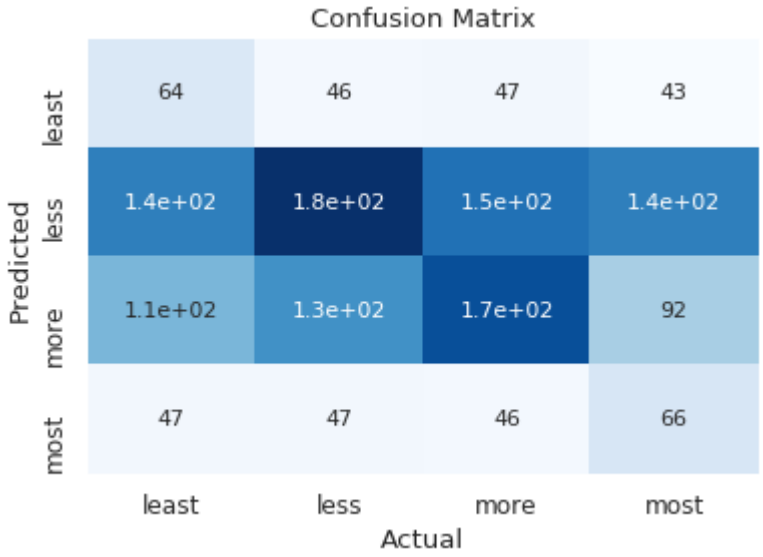
```
confusion = learner.evaluate()
```

```
              precision    recall  f1-score   support

           0       0.18      0.32      0.23       200
           1       0.45      0.30      0.36       618
           2       0.41      0.34      0.37       501
           3       0.19      0.32      0.24       206

    accuracy                           0.32      1525
   macro avg       0.31      0.32      0.30      1525
weighted avg       0.37      0.32      0.33      1525
```

In [14]:

```
# 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)
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 [ ]: