Metric: Accuracy

ULMFiT - 75.3679%

Language Model ULMFiT Performance Metric

```
[ ]  1 learn.fit_one_cycle(2, min_grad_lr)
```

epoch	train_loss	valid_loss	accuracy	time
0	7.023629	6.735806	0.111317	04:04
1	6.551484	6.556456	0.119150	04:05

We can do a few more epochs after unfreezing all the layers.

```
[ ]  1 learn.unfreeze()
     2 learn.fit_one_cycle(2, 1e-3)
```

epoch	train_loss	valid_loss	accuracy	time
0	6.180114	6.442215	0.124212	04:31
1	5.969887	6.409162	0.126822	04:32

Classifier Model ULMFiT Performance Metric

```
[ ]  1 learn.unfreeze()
     2 learn.fit_one_cycle(4, slice(2e-3/100, 2e-3), moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.769100	0.782119	0.749474	03:11
1	0.755485	0.773947	0.751777	02:55
2	0.727611	0.774901	0.753012	02:38
3	0.702299	0.781336	0.753679	02:54

Model Evaluation

```
 1 preds,tensor,probs=learn.predict("football is really nice")
 2 top_pred = probs.argsort(descending=True)[:1]
 3 classes=learn.data.classes
 4 labels = []
 5 confidence = []
 6 for i in top_pred:
 7   x = classes[i]
 8   p = probs[i]
 9   labels.append(x)
10   print("The following sentence belongs to", labels, "with confidence as", p)
```

```
The following sentence belongs to ['Sports'] with confidence as tensor(0.9767)
```

```
[ ]  1 preds,tensor,probs=learn.predict("homebrew is not working on macosx")
     2 top_pred = probs.argsort(descending=True)[:1]
     3 classes=learn.data.classes
     4 labels = []
     5 confidence = []
     6 for i in top_pred:
     7   x = classes[i]
     8   p = probs[i]
     9   labels.append(x)
    10   print("The following sentence belongs to", labels, "with confidence as", p)
```

```
The following sentence belongs to ['Education & Reference'] with confidence as tensor(0.3505)
```

BERT - 81.93%

BERT Model Performance Metric

```
[ ]  1 #@ Fitting model using one-cycle poliy, lr=2e-5 from research paper
     2 learner.fit_onecycle(lr=2e-5, epochs=3);
```

```
begin training using onecycle policy with max lr of 2e-05...
Epoch 1/3
1405/1405 [==============================] - 835s 594ms/step - loss: 0.9965 - accuracy: 0.6873 - val_loss: 0.7922 - val_accuracy: 0.7485
Epoch 2/3
1405/1405 [==============================] - 828s 589ms/step - loss: 0.7199 - accuracy: 0.7716 - val_loss: 0.7388 - val_accuracy: 0.7640
Epoch 3/3
1405/1405 [==============================] - 824s 586ms/step - loss: 0.5698 - accuracy: 0.8193 - val_loss: 0.7567 - val_accuracy: 0.7656
```

Model Evaluation

```
[ ]  1 data = ["football is really nice"]
```

```
     1 predictor.predict(data)
```

```
['Sports']
```

```
[ ]  1 data = ['football is really nice']
     2 classes=predictor.predict(data)
     3 probs=predictor.predict(data, return_proba=True)
     4 # probs = torch.Tensor(probs)
     5 # top_pred = probs.argsort(descending=True)[:,0]
     6 print("The following sentence belongs to", classes, "with confidence:", probs.max())
```

```
The following sentence belongs to ['Sports'] with confidence: 0.97864574
```

Example 2.

```
[ ]  1 data = ['homebrew is not working on macosx']
```

```
[ ]  1 predictor.predict(data)
```

```
['Computers & Internet']
```

```
[ ]  1 data = ['homebrew is not working on macosx, can someone solve this problem?']
     2 classes=predictor.predict(data)
     3 probs=predictor.predict(data, return_proba=True)
     4 # probs = torch.Tensor(probs)
     5 # top_pred = probs.argsort(descending=True)[:,0]
     6 print("The following sentence belongs to", classes, "with confidence:", probs.max())
```

```
The following sentence belongs to ['Computers & Internet'] with confidence: 0.9726442
```