bert

April 15, 2021

1 Emotion Classification in short texts with BERT

Applying BERT to the problem of multiclass text classification. Our dataset consists of written dialogs, messages and short stories. Each dialog utterance/message is labeled with one of the five emotion categories: joy, anger, sadness, fear, neutral.

1.1 Workflow:

1. Import Data
2. Data preprocessing and downloading BERT
3. Training and validation
4. Saving the model

Multiclass text classification with BERT and ktrain. Use google colab for a free GPU

Let's start

[ ]:
```
# install ktrain on Google Colab
!pip3 install ktrain
```

[2]:
```
import pandas as pd
import numpy as np

import ktrain
from ktrain import text
```

```
<IPython.core.display.HTML object>
```

```
using Keras version: 2.2.4-tf
```

1.2 1. Import Data

[4]:
```
data_train = pd.read_csv('data/data_train.csv', encoding='utf-8')
data_test = pd.read_csv('data/data_test.csv', encoding='utf-8')

X_train = data_train.Text.tolist()
X_test = data_test.Text.tolist()

y_train = data_train.Emotion.tolist()
y_test = data_test.Emotion.tolist()
```

```
data = data_train.append(data_test, ignore_index=True)

class_names = ['joy', 'sadness', 'fear', 'anger', 'neutral']

print('size of training set: %s' % (len(data_train['Text'])))
print('size of validation set: %s' % (len(data_test['Text'])))
print(data.Emotion.value_counts())

data.head(10)
```

```
size of training set: 7934
size of validation set: 3393
joy         2326
sadness     2317
anger       2259
neutral     2254
fear        2171
Name: Emotion, dtype: int64
```

[4]:
```
   Emotion                                               Text
0  neutral   There are tons of other paintings that I thin...
1  sadness  Yet the dog had grown old and less capable , a...
2     fear  When I get into the tube or the train without ...
3     fear  This last may be a source of considerable disq...
4    anger  She disliked the intimacy he showed towards so...
5  sadness  When my family heard that my Mother's cousin w...
6      joy  Finding out I am chosen to collect norms for C...
7    anger  A spokesperson said : ` Glen is furious that t...
8  neutral                                             Yes .
9  sadness  When I see people with burns I feel sad, actua...
```

[ ]:
```
encoding = {
    'joy': 0,
    'sadness': 1,
    'fear': 2,
    'anger': 3,
    'neutral': 4
}

# Integer values for each class
y_train = [encoding[x] for x in y_train]
y_test = [encoding[x] for x in y_test]
```

1.3 2. Data preprocessing

- The text must be preprocessed in a specific way for use with BERT. This is accomplished by setting preprocess_mode to 'bert'. The BERT model and vocabulary will be automatically downloaded
- BERT can handle a maximum length of 512, but let's use less to reduce memory and improve speed.

[6]:
```
(x_train,  y_train), (x_test, y_test), preproc = text.
↪texts_from_array(x_train=X_train, y_train=y_train,
                                                                    ␣
↪x_test=X_test, y_test=y_test,
                                                                    ␣
↪class_names=class_names,
                                                                    ␣
↪preprocess_mode='bert',
                                                                    ␣
↪maxlen=350,
                                                                    ␣
↪max_features=35000)
```

```
downloading pretrained BERT model (uncased_L-12_H-768_A-12.zip)...
[]
extracting pretrained BERT model...
done.

cleanup downloaded zip...
done.

preprocessing train...
language: en

<IPython.core.display.HTML object>

preprocessing test...
language: en

<IPython.core.display.HTML object>
```

1.4 2. Training and validation

Loading the pretrained BERT for text classification

[7]:
```
model = text.text_classifier('bert', train_data=(x_train, y_train),␣
↪preproc=preproc)
```

```
Is Multi-Label? False
maxlen is 350
done.
```

Wrap it in a Learner object

```
[ ]: learner = ktrain.get_learner(model, train_data=(x_train, y_train),
                                  val_data=(x_test, y_test),
                                  batch_size=6)
```

Train the model. More about tuning learning rates here

```
[9]: learner.fit_onecycle(2e-5, 3)
```

```
begin training using onecycle policy with max lr of 2e-05...
Train on 7934 samples, validate on 3393 samples
Epoch 1/3
7934/7934 [==============================] - 475s 60ms/sample - loss: 0.9311 -
acc: 0.6364 - val_loss: 0.5669 - val_acc: 0.8034
Epoch 2/3
7934/7934 [==============================] - 466s 59ms/sample - loss: 0.4569 -
acc: 0.8470 - val_loss: 0.5211 - val_acc: 0.8232
Epoch 3/3
7934/7934 [==============================] - 466s 59ms/sample - loss: 0.1911 -
acc: 0.9411 - val_loss: 0.5589 - val_acc: 0.8320
```

```
[9]: <tensorflow.python.keras.callbacks.History at 0x7ffa776ace10>
```

Validation

```
[10]: learner.validate(val_data=(x_test, y_test), class_names=class_names)
```

```
              precision    recall  f1-score   support

         joy       0.87      0.85      0.86       707
     sadness       0.84      0.79      0.82       676
        fear       0.86      0.87      0.86       679
       anger       0.81      0.80      0.81       693
     neutral       0.78      0.85      0.81       638

    accuracy                           0.83      3393
   macro avg       0.83      0.83      0.83      3393
weighted avg       0.83      0.83      0.83      3393
```

```
[10]: array([[598,   8,  15,  13,  73],
             [ 18, 537,  37,  54,  30],
             [ 16,  20, 590,  40,  13],
             [ 19,  49,  35, 557,  33],
             [ 37,  24,  12,  24, 541]])
```

Testing with other inputs

```
[11]: predictor = ktrain.get_predictor(learner.model, preproc)
      predictor.get_classes()
```

```
[11]: ['joy', 'sadness', 'fear', 'anger', 'neutral']
```

```
[16]: import time

      message = 'I just broke up with my boyfriend'

      start_time = time.time()
      prediction = predictor.predict(message)

      print('predicted: {} ({:.2f})'.format(prediction, (time.time() - start_time)))
```

```
<IPython.core.display.HTML object>

predicted: sadness (0.06)
```

1.5 4. Saving Bert model

```
[ ]: # let's save the predictor for later use
     predictor.save("models/bert_model")
```

Done! to reload the predictor use: ktrain.load_predictor