Documentation

Course Project: Text Classification – Sarcasm Detection

Problem: Detection of sarcasm from tweets. I got set of training list of tweets and the context for each tweet. using the state of art text classification practices I need to generate a model to solve the problem.

Approach: Using Tensorflow2 and Keras deep learning API with Pretrained BERT to generate a model.

Results: I have tried multiple models with various combination with BERT,LSTM and finally following model gave me better results.

Below are a validation results for 500 tweets, Validated from the generated trained model(Note: results varies by training epochs).

	precision	recall	f1-score	support
SARCASM	0.79	0.80	0.80	250
NOT_SARCASM	0.80	0.79	0.80	250
accuracy			0.80	500
macro avg	0.80	0.80	0.80	500
weighted avg	0.80	0.80	0.80	500

BERT (Bidirectional Encoder Representations from Transformers):

BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. It was wildly successful on a variety of tasks in NLP (natural language processing). They compute vector-space representations of natural language that are suitable for use in deep learning models. The BERT family of models uses the Transformer encoder architecture to process each token of input text in the full context of all tokens before and after, hence the name: Bidirectional Encoder Representations from Transformers. More details are available at https://github.com/google-research/bert
In our model I used L=12 hidden layers (i.e., Transformer blocks), a hidden size of H=768, and A=12 attention heads. Available at
https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip

Using the BERT model created a training model:

```python
def create_model(max_seq_len, bert_file):

  with tf.io.gfile.GFile(bert_config, "r") as reader:
      bc = StockBertConfig.from_json_string(reader.read())
      bert_params = map_stock_config_to_params(bc)
      bert_params.adapter_size = None
      bert = BertModelLayer.from_params(bert_params, name="bert")

  input_ids = keras.layers.Input(shape=(max_seq_len, ), dtype='int32', name="input_ids")
  bert_output = bert(input_ids)

  cls_out = keras.layers.Lambda(lambda seq: seq[:, 0, :])(bert_output)
  cls_out = keras.layers.Dropout(0.5)(cls_out)
  logits = keras.layers.Dense(units=250, activation="tanh")(cls_out)
  logits = keras.layers.Dropout(0.5)(logits)
  logits = keras.layers.Dense(units=2, activation="sigmoid")(logits)

  model = keras.Model(inputs=input_ids, outputs=logits)

  model.build(input_shape=(None, max_seq_len))
  model.summary()
  load_stock_weights(bert, bert_file)

  return model
```

Summary of model:

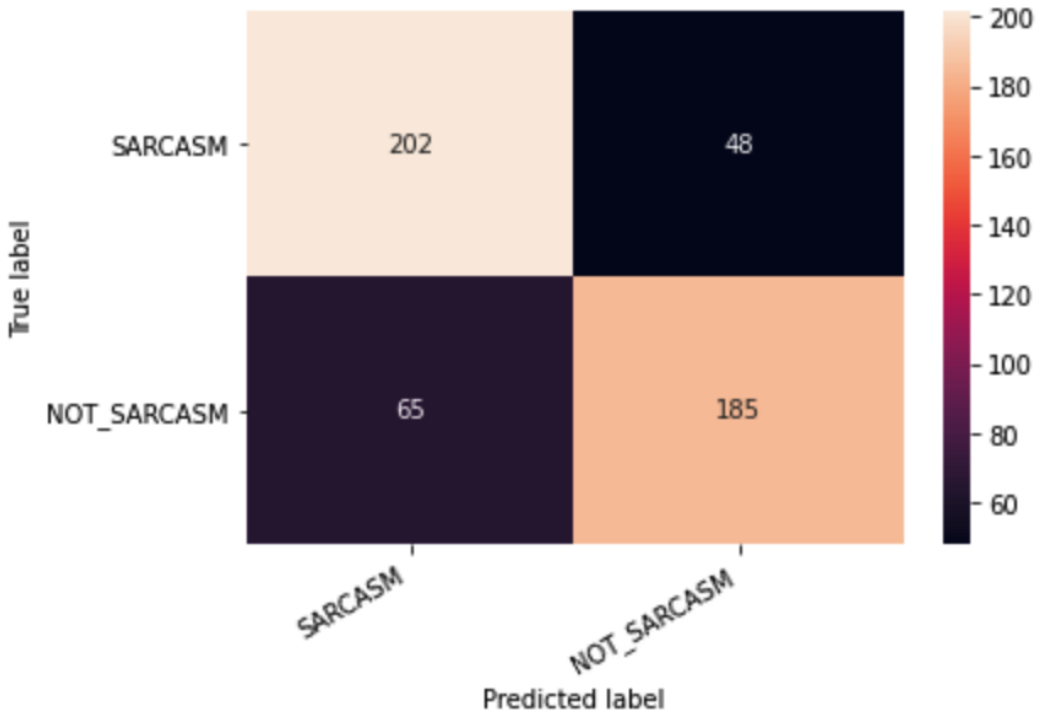
```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_ids (InputLayer)       [(None, 85)]              0
_________________________________________________________________
bert (BertModelLayer)        (None, 85, 768)           108890112
_________________________________________________________________
lambda_1 (Lambda)            (None, 768)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 768)               0
_________________________________________________________________
dense_2 (Dense)              (None, 250)               192250
_________________________________________________________________
dropout_3 (Dropout)          (None, 250)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 502
=================================================================
Total params: 109,082,864
Trainable params: 109,082,864
Non-trainable params: 0
```

Classification Report:

```
[ ] print(classification_report(data.test_y, y_pred, target_names=classes))
```

	precision	recall	f1-score	support
SARCASM	0.76	0.81	0.78	250
NOT_SARCASM	0.79	0.74	0.77	250
accuracy			0.77	500
macro avg	0.78	0.77	0.77	500
weighted avg	0.78	0.77	0.77	500

Confusion Matrix:



With this model, we can clearly see the accuracy after the training reached 77 percentage. I got the similar results for various fine-tuning efforts. As part of finetuning I have updated the training learning rate of optimizer and epsilon also tried other activation logics for layers in the models.

Other Model: BERT + LSTM

```
def create_model_lstm(max_seq_len, bert_file):

  with tf.io.gfile.GFile(bert_config, "r") as reader:
      bc = StockBertConfig.from_json_string(reader.read())
      bert_params = map_stock_config_to_params(bc)
      bert_params.adapter_size = None
      bert = BertModelLayer.from_params(bert_params, name="bert")

  input_ids = keras.layers.Input(shape=(max_seq_len, ), dtype='int32', name="input_ids")
  bert_output = bert(input_ids)
  lstm = keras.layers.LSTM(units=768,return_sequences=True,name="LSTM")(bert_output)
  print("bert shape", bert_output.shape)

  cls_out = keras.layers.Lambda(lambda seq: seq[:, 0, :])(lstm)
  cls_out = keras.layers.Dropout(0.5)(cls_out)
  logits = keras.layers.Dense(units=768, activation="tanh")(cls_out)
  logits = keras.layers.Dropout(0.5)(logits)
  logits = keras.layers.Dense(units=len(classes), activation="softmax")(logits)

  model = keras.Model(inputs=input_ids, outputs=logits)
  model.build(input_shape=(None, max_seq_len))
  model.summary()
  load_stock_weights(bert, bert_file)

  return model
```

```
[17] model = create_model_lstm(data.max_seq_len, bert_file)
```

```
bert shape (None, 70, 768)
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_ids (InputLayer)       [(None, 70)]              0
_________________________________________________________________
bert (BertModelLayer)        (None, 70, 768)           108890112
_________________________________________________________________
LSTM (LSTM)                  (None, 70, 768)           4721664
_________________________________________________________________
lambda (Lambda)              (None, 768)               0
_________________________________________________________________
dropout (Dropout)            (None, 768)               0
_________________________________________________________________
dense (Dense)                (None, 768)               590592
_________________________________________________________________
dropout_1 (Dropout)          (None, 768)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1538
=================================================================
Total params: 114,203,906
Trainable params: 114,203,906
Non-trainable params: 0
_________________________________________________________________
Done loading 196 BERT weights from: model/uncased_L-12_H-768_A-12/bert_model.ckpt into <
Unused weights from checkpoint:
        bert/embeddings/token_type_embeddings
        bert/pooler/dense/bias
        bert/pooler/dense/kernel
        cls/predictions/output_bias
        cls/predictions/transform/LayerNorm/beta
        cls/predictions/transform/LayerNorm/gamma
        cls/predictions/transform/dense/bias
        cls/predictions/transform/dense/kernel
        cls/seq_relationship/output_bias
        cls/seq_relationship/output_weights
```

Classification Report:

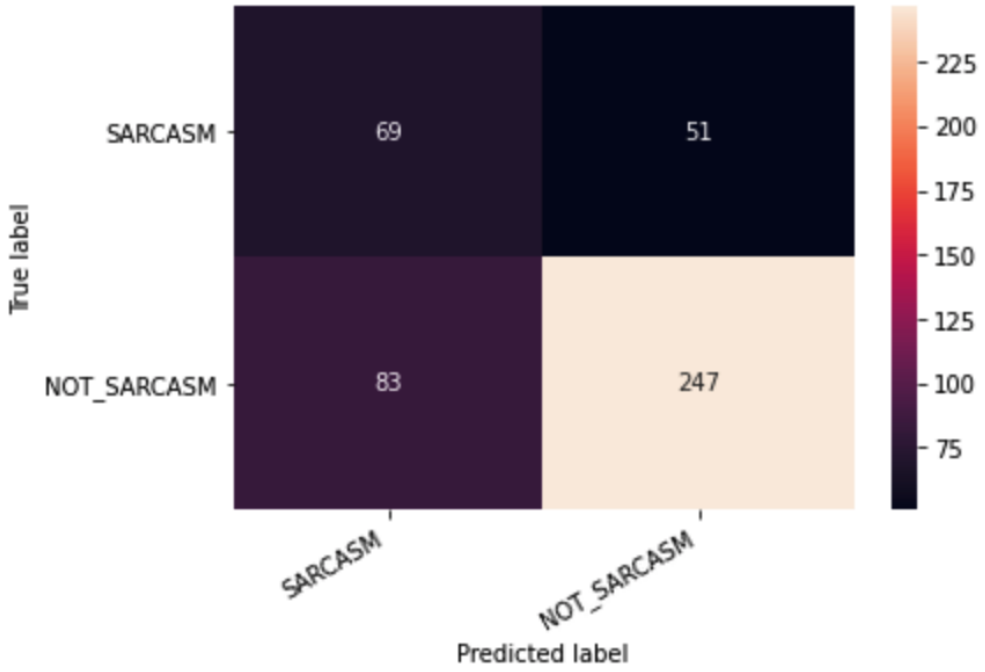
```
print(classification_report(data.test_y, y_pred, target_names=classes))
```

```
              precision    recall  f1-score   support

     SARCASM       0.45      0.57      0.51       120
NOT_SARCASM       0.83      0.75      0.79       330

    accuracy                           0.70       450
   macro avg       0.64      0.66      0.65       450
weighted avg       0.73      0.70      0.71       450
```

Confusion Matrix:



Clearly the LSTM model doesn't do as expected. The accuracy doesn't reached the limit set for the competition.

Resources:

- https://github.com/google-research/bert
- https://www.tensorflow.org/tutorials/text/classify_text_with_bert
- https://towardsdatascience.com/bert-for-dummies-step-by-step-tutorial-fb90890ffe03
- https://medium.com/atheros/text-classification-with-transformers-in-tensorflow-2-bert-2f4f16eff5ad