BERT

資管三 A 黃彥程 109403526

Accuracy:

0.9128

```
[ ]  correct = 0
     for idx, pred in enumerate(res['pred']):
       if pred == res['label'][idx]:
         correct += 1
     print('test accuracy = %.4f'%(correct/len(test_df)))

     test accuracy = 0.9128
```

網址:

BERT 0.9128

https://colab.research.google.com/drive/1J0meDYpPV0NuK5G2eU46uThUDjX1jPkO

DistilBERT 0.6546

https://colab.research.google.com/drive/17YlzcSjSNney9OgN6PeLddTSEwtaJ0ia

這種 BERT 的變形比較小巧，可以在有限資源內取得更好的成效。

RoBERTa 0.5284

https://colab.research.google.com/drive/1E-bwD0e58bVmcJmK6I6eSLaGPPwBGpIK

這種 BERT 變形比較厚實，可以比 BERT 有更好的效能，但所使用之

模型較大，需要的資源較多。

過程:

Todo1&2

在做之前，雖然有看過助教提供的範本，但由於之前除了寫這堂課第一次作業之外，沒碰過機器學習程式碼，再說我多看幾次也覺得這次作業跟上次的可以說八竿子打不著，整個架構都不一樣，上次作業可以線性按照助教給的提示往下做(雖然也是查很久怎麼寫)，但這次作業我看到前兩個 todo，頓時間不知道怎麼反應，硬是往下找這兩個函式會在哪裡用到，上下來回比對，才稍微比較了解，圖片作業中是我後來看到範本有一樣的動作所以就參考過來了。

Call_metric 的部分則是寫到 tokenizer 跟 evaluate 時看到需要回傳那些參數(各種 score)，才回頭去查然後寫出來。

```python
# get predict result
def get_pred(logits):

    return logits.argmax(dim=1)                          # todo #

# calculate confusion metrics
def cal_metrics(pred, ans):

  pred = pred.cpu().numpy()
  ans = ans.cpu().numpy()

  acc = accuracy_score(ans, pred)
  f1 = f1_score(ans, pred)
  recall = recall_score(ans, pred)            # todo #
  precision = precision_score(ans, pred)

  return acc, f1, recall, precision
```

Todo 3

用 append 把他們連上

- TODO3:把資料拿出來後，將train及test合併，重新切割後，儲存下來。

```python
import pandas as pd

all_df = [] # a list to save all data

all_df = pd.DataFrame(dataset['train']).append(pd.DataFrame(dataset['test']))   # todo #
all_df.head()
```

	text	label
0	I rented I AM CURIOUS-YELLOW from my video sto...	0
1	"I Am Curious: Yellow" is a risible and preten...	0
2	If only to avoid making this type of film in t...	0
3	This film was probably inspired by Godard's Ma...	0
4	Oh, brother...after hearing about this ridicul...	0

檢查是否正確

```
0    0.5
1    0.5
Name: label, dtype: float64
```

```
# of train_df: 40000
# of dev_df: 5000
# of test_df data: 5000
```

Todo 4

依 bert-base-uncased 設置好的 tokenizer 格式會回傳 input_ids, attention_mask, token_type_ids。

```
# transform text to its number
def tokenize(self,input_text):
  result = self.tokenizer.encode_plus(
      input_text,
      add_special_tokens=True,
      max_length=self.max_len,
      padding='max_length',
      return_token_type_ids=True,
      truncation=True,
  )
  input_ids = result["input_ids"]
  attention_mask = result["attention_mask"]
  token_type_ids = result["token_type_ids"]

  return input_ids, attention_mask, token_type_ids


# get single data
```

Todo 5

按照助教給的提示依序加入，除了

bt_output = bert_output.pooler_output

這行是我在查資料時發現大家都有寫到的。

```python
# BERT Model
class BertClassifier(BertPreTrainedModel):
  def __init__(self, config, args):
    super(BertClassifier, self).__init__(config)
    self.bert = BertModel(config)

    self.dropout = nn.Dropout(args["dropout"])
    self.classifier = nn.Linear(config.hidden_size, args["num_class"])  # todo #

    self.init_weights()

  # forward function, data in model will do this
  def forward(self, input_ids=None, attention_mask=None, token_type_ids=None, position_ids=None,
              head_mask=None, inputs_embeds=None, labels=None, output_attentions=None,
              output_hidden_states=None, return_dict=None):
    bert_output = self.bert(input_ids, attention_mask=attention_mask, token_type_ids=token_type_ids,
                            position_ids=position_ids, head_mask=head_mask, inputs_embeds=inputs_embeds,
                            output_attentions=output_attentions, output_hidden_states=output_hidden_states,
                            return_dict=return_dict)
    bt_output = bert_output.pooler_output
    output = self.dropout(bt_output)
    logits = self.classifier(output)

                                    # todo #

    return logits
```

Hyperparameter

在 train model 時自己嘗試 learning rate, len ，那裏寫了 negative 是

我在嘗試時提醒自己就先別動 dropout=0.4。

```python
from datetime import datetime
parameters = {
    "num_class": 2,
    "time": str(datetime.now()).replace(" ", "_"),
    # Hyperparameters
    "model_name": 'BERT',
    "config": 'bert-base-uncased',
    "learning_rate": 1e-6,
    "epochs": 3,
    "max_len": 512,
    "batch_size": 16,
    "dropout": 0.4,  # negative
}
```

Todo 6

參照 evaluate 寫出訓練方式。

```python
for epoch in range(parameters["epochs"]):

    st_time = time.time()
    train_loss, train_acc, train_f1, train_rec, train_prec = 0.0, 0.0, 0.0, 0.0, 0.0
    step_count = 0

    model.train()
    for data in train_loader:
        ids, masks, token_type_ids, labels = [t.to(device) for t in data]

        optimizer.zero_grad()
        logits = model(input_ids = ids,
              token_type_ids = token_type_ids,
              attention_mask = masks)

        acc, f1, rec, prec = cal_metrics(get_pred(logits), labels)
        loss = loss_fct(logits, labels)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        train_acc += acc
        train_f1 += f1
        train_rec += rec
        train_prec += prec
        step_count += 1
```

一段時間過去了↓

```
[epoch 1] cost time: 366.4092 s
          loss     acc      f1       rec     prec
train | 0.0813, 0.9780, 0.9761, 0.9868, 0.9685
val   | 0.2270, 0.9199, 0.9213, 0.9480, 0.9093

[epoch 2] cost time: 371.5665 s
          loss     acc      f1       rec     prec
train | 0.0832, 0.9790, 0.9776, 0.9857, 0.9724
val   | 0.2103, 0.9258, 0.9245, 0.9421, 0.9181

[epoch 3] cost time: 371.1287 s
          loss     acc      f1       rec     prec
train | 0.0856, 0.9760, 0.9737, 0.9851, 0.9658
val   | 0.2337, 0.9199, 0.9243, 0.9423, 0.9169
```

Todo 7

裡面的'bert-base-uncased'寫 parameter['config']可能比較好，我忘記改了，不過也是一樣的東西不影響。

```
[ ]  # predict single sentence, return each-class's probability and predicted class
     def predict_one(query, model):
       tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')

       inputs = tokenizer.encode_plus(
             query,
             add_special_tokens=True,
             max_length = parameters["max_len"],
             truncation = True,
             padding = 'max_length',
             return_token_type_ids=True
         )
       query_ids = torch.tensor([inputs.input_ids], dtype=torch.long).to(device)
       token_type_ids = torch.tensor([inputs.token_type_ids], dtype=torch.long).to(device)
       masks = torch.tensor([inputs.attention_mask], dtype=torch.long).to(device)


       logits = model(input_ids = query_ids,
                   token_type_ids = token_type_ids,
                   attention_mask = masks)

       probs = logits
       pred = get_pred(logits).cpu().numpy().item()

       # todo #
       ##########

       return probs, pred
```

心得:

首先就是成就感，上次就覺得作業內容蠻有趣充實，這次雖然看上去比上次亂，個人感覺也比較難，寫之前心裡沒有一個底這作業大概長怎樣要怎麼寫，但不停查資料慢慢把內容寫出來，然後逐漸理解整個到底怎麼運作，還是很有趣很有成就感的，然後就是一些過程，那些東西都寫出來之後，就要開始調整參數(Hyperparameter)去 train 寫好的 model，剛開始概念都不清楚，參數可以說是在抽樂透，有幾次跑到 80 以上，但我又覺得不夠，多測幾次沒有意外果然是卡回 50 左右，就好像在說:「這模型跟我一樣，學不會」，後來想

說不行，我還是去了解一下那些參數的影響，最後是試出了還可以的分數。

至於其他兩個模型，比較值得注意的是，DistilBERT 這個模型的 tokenizer 是沒有回傳 token_type_id 的，我想是因為他這個模型本身的目的就是為了迎合當資源有限時能夠更有效的達到目標。

而 RoBERTa 我有發現到同樣的 batch size 他要使用的 learning rate 比較低，不然 recall_score 跟 f1_score 會一起 0.0000，要到 1e-7~1e-9 才會比較有結果。

做完本次作業我有感覺到我數據整理的能力，跟對機器學習有顯著得提升，雖然我知道這些都還只是一些很淺的內容，不過確實有收穫!