Báo Cáo Bert và Distilbert

1. Tạo bộ dữ liệu train, valid, test:

* 60% dữ liệu đầu cho training.
* 20% dữ liệu tiếp theo cho validation.
* 20% dữ liệu cuối cùng cho testing.

1. Xây dựng mô hình với số nhãn là 2:

* Mô hình Bert: 'bert-base-uncased'.
* Mô hình XLNet: 'xlnet-base-cased'.

1. Class Tokenize:

class Tokenizer:

def \_\_init\_\_(self, model\_name, max\_length=128):

if 'bert' in model\_name:

self.tokenizer = BertTokenizerFast.from\_pretrained(model\_name)

 elif 'xlnet' in model\_name:

      self.tokenizer = XLNetTokenizerFast.from\_pretrained(model\_name)

  self.max\_length = max\_length

def tokenize\_function(self, examples):

return self.tokenizer(

  examples["text"],

      truncation=True,

      padding="max\_length",

      max\_length=self.max\_length

  )

1. Thiết lập Hyperparameter cho training:

* Output\_dir=’./bert\_results’. # Lưu kết quả huấn luyện vào file bert\_results.
* per\_device\_train\_batch\_size=8.
* per\_device\_eval\_batch\_size=8. # Thiết lập kích thước batch cho tập huấn luyện và tập đánh giá.
* num\_train\_epochs=1 # Số lần mô hình huấn luyện trên toàn bộ tập dữ liệu.
* weight\_decay=0.01 # Áp dụng kỹ thuật điều chuẩn L2 để giúp mô hình tránh overfitting.
* evaluation\_strategy="epoch" # Tần suất đánh giá mô hình.
* save\_strategy="epoch" # Tần suất lưu loss, learning\_rate, grad\_norm, trọng số của mô hình.

1. Huấn luyện mô hình:

* Sử dụng các thông số như accuracy, f1-score, recall, precision-score để đánh giá.

1. Đánh giá mô hình:

* Bert model:
* On train data:

{'eval\_loss': 0.1548750251531601, 'eval\_accuracy': 0.9452584007580728, 'eval\_f1': 0.945269902401505, 'eval\_precision': 0.9453140999778809, 'eval\_recall': 0.9452584007580728, 'eval\_confusion\_matrix': [[6830, 404], [347, 6138]], 'eval\_runtime': 88.7281, 'eval\_samples\_per\_second': 154.618, 'eval\_steps\_per\_second': 19.329, 'epoch': 1.0}

{'train\_runtime': 783.1495, 'train\_samples\_per\_second': 52.554, 'train\_steps\_per\_second': 3.284, 'train\_loss': 0.24195206962514257, 'epoch': 1.0}

* On test data:

{'test\_loss': 0.1624840348958969, 'test\_accuracy': 0.9426384839650146, 'test\_f1': 0.942699935414346, 'test\_precision': 0.9429619844999592, 'test\_recall': 0.9426384839650146, 'test\_confusion\_matrix': [[7174, 464], [323, 5759]], 'test\_runtime': 79.4158, 'test\_samples\_per\_second': 172.761, 'test\_steps\_per\_second': 21.595}

* Distilbert model:
* On train data:

{'eval\_loss': 0.1640801578760147, 'eval\_accuracy': 0.9394270719440192, 'eval\_f1': 0.9394599655808152, 'eval\_precision': 0.9398129148566099, 'eval\_recall': 0.9394270719440192, 'eval\_confusion\_matrix': [[6729, 505], [326, 6159]], 'eval\_runtime': 59.4362, 'eval\_samples\_per\_second': 230.819, 'eval\_steps\_per\_second': 28.854, 'epoch': 1.0}

{'train\_runtime': 471.5333, 'train\_samples\_per\_second': 87.285, 'train\_steps\_per\_second': 5.455, 'train\_loss': 0.23344826957856885, 'epoch': 1.0}

* On test data:

{'test\_loss': 0.16996903717517853, 'test\_accuracy': 0.9381195335276968, 'test\_f1': 0.9382232555685888, 'test\_precision': 0.9388897067300341, 'test\_recall': 0.9381195335276968, 'test\_confusion\_matrix': [[7095, 543], [306, 5776]], 'test\_runtime': 53.5045, 'test\_samples\_per\_second': 256.427, 'test\_steps\_per\_second': 32.053}

1. Kết luận:

* Mô hình Bert cho ra các thông số đánh giá tốt hơn nhưng thời gian huấn luyện lâu hơn gấp đôi