BERT for multiple NLP Tasks: Telecom Data

Lab Objectives

This lab focuses on applying BERT (Bidirectional Encoder Representations from Transformers) to analyze telecom customer service conversations. Through this practical session, you will learn how to leverage BERT for various NLP tasks specifically in the telecom domain.

Dataset Description

The dataset contains customer service conversations between users and telecom company agents.

Data Dictionary:

- `sms` : SMS message
- `label` : 0 or 1

Tasks Overview

1. **Data Preparation and Exploration**
   - Load and examine the conversation dataset
   - Perform basic text preprocessing
   - Prepare data for BERT input
2. **Sentence Similarity Analysis**
   - Implement BERT embeddings for sentences
   - Create similarity matrices for customer queries
   - Build a query matching system
3. **Document Clustering**
   - Generate BERT embeddings for entire conversations
   - Implement clustering algorithms
   - Analyze and visualize conversation clusters
4. **Text Classification**
   - Fine-tune BERT for category prediction
   - Evaluate classification performance
   - Analyze misclassified cases

```
# ! pip install transformers --quiet
```

Importing libraries

```
# Required imports
import torch
from transformers import BertTokenizer, BertModel, BertForSequenceClassification
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import cosine_similarity
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import warnings

warnings.filterwarnings('ignore')
```

Dataset file

```
# Load the dataset
df = pd.read_csv('/content/SMS_spam_detection_train.csv')

# Display basic information about the dataset
print("Dataset Shape:", df.shape)
print("\nDataset Columns:", df.columns.tolist())
```

```
Dataset Shape: (5574, 2)

Dataset Columns: ['sms', 'label']
```

```
df.head()
```

	sms	label
0	Go until jurong point, crazy.. Available only ...	0
1	Ok lar... Joking wif u oni...\n	0
2	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	U dun say so early hor... U c already then say...	0
4	Nah I don't think he goes to usf, he lives aro...	0

Text processing function

```
def preprocess_text(text):
    """Basic preprocessing for BERT input"""
```

```
    # Convert to string in case of missing values
    text = str(text)

    # Remove extra whitespace
    text = ' '.join(text.split())

    # Truncate to BERT's maximum sequence length (512 tokens)
    return text[:512]


# Apply preprocessing
df['process_text'] = df['sms'].apply(preprocess_text)
```

Loading BERT model

```
# Initialize tokenizer and model
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

# Move model to GPU if available
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
```

```
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 3.16kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 8.89MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 2.70MB/s]
config.json: 100%  570/570 [00:00<00:00, 33.5kB/s]
model.safetensors: 100%  440M/440M [00:02<00:00, 248MB/s]
```

Sentence Similarity Analysis

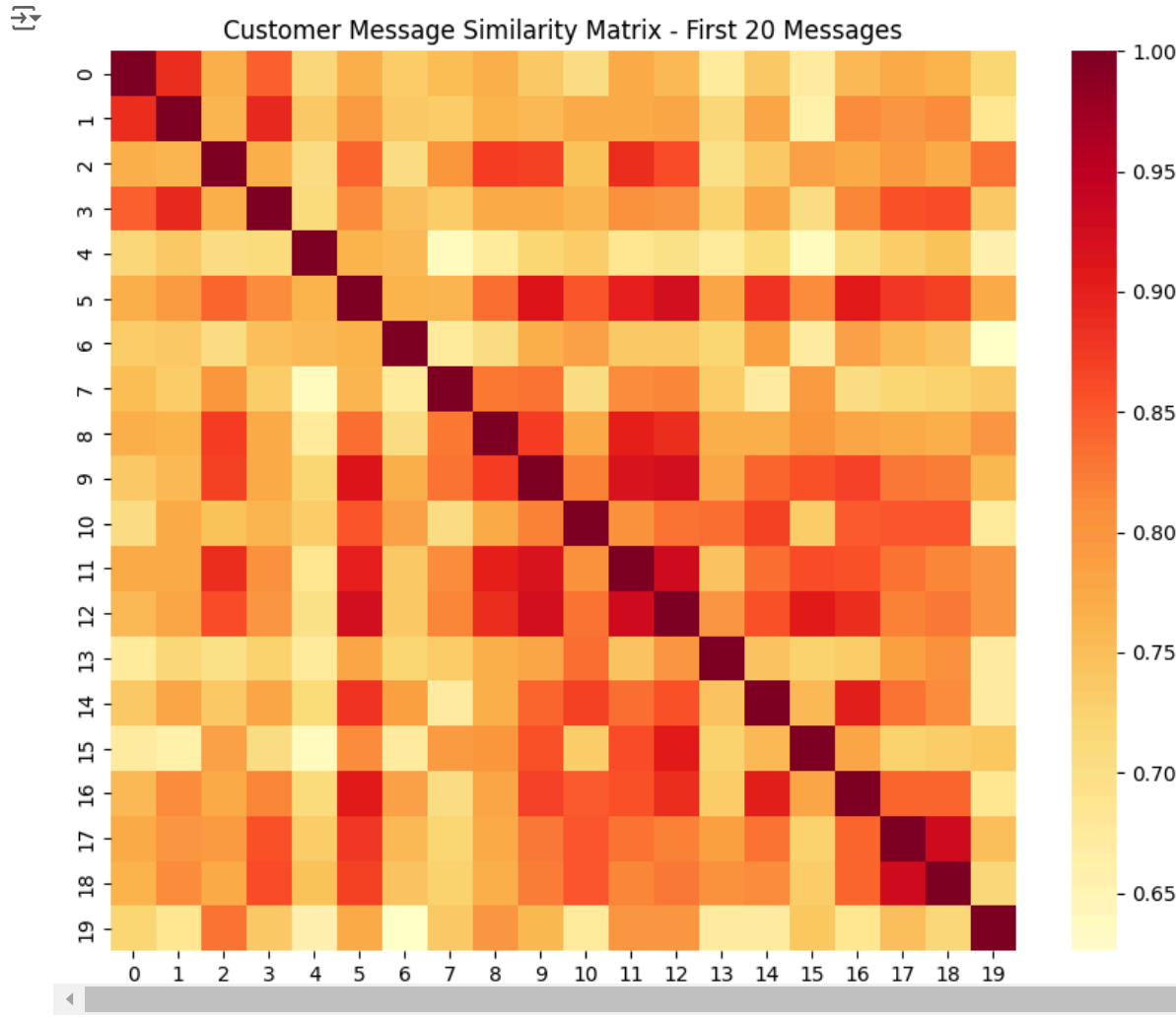
Embdding function

```
def get_bert_embedding(text, tokenizer, model):
    """Generate BERT embedding for a given text"""
    # Tokenize and convert to tensor
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True)
    inputs = {k: v.to(device) for k, v in inputs.items()}

    # Generate embeddings
    with torch.no_grad():
        outputs = model(**inputs)

    # Use [CLS] token embedding as sentence representation
    return outputs.last_hidden_state[:, 0, :].cpu().numpy()
```

Functionfor similarity calculation

```
def calculate_similarity_matrix(messages):
    """Calculate similarity matrix for a list of messages"""
    embeddings = []
    for msg in messages:
        emb = get_bert_embedding(msg, tokenizer, model)
        embeddings.append(emb[0])

    # Calculate cosine similarity
    similarity_matrix = cosine_similarity(embeddings)
    return similarity_matrix



# Generate similarity matrix for a sample of customer messages
sample_messages = df['process_text'].head(100)
similarity_matrix = calculate_similarity_matrix(sample_messages)

# Visualize similarity matrix
plt.figure(figsize=(10, 8))
sns.heatmap(similarity_matrix, cmap='YlOrRd')
plt.title('Customer Message Similarity Matrix')
plt.show()
```


```
# Visualize similarity matrix
plt.figure(figsize=(10, 8))
sns.heatmap(similarity_matrix[:20, :20], cmap='YlOrRd')
plt.title('Customer Message Similarity Matrix - First 20 Messages')
plt.show()
```



Document Clustering

```
# Generate embeddings for clustering
conversation_embeddings = []

for msg in df["process_text"]:  # Using a subset for demonstration
    emb = get_bert_embedding(msg, tokenizer, model)
    conversation_embeddings.append(emb[0])


K = 5
kmeans = KMeans(n_clusters=K,random_state=42).fit(conversation_embeddings)
cls_dist=pd.Series(kmeans.labels_).value_counts()


cls_dist
```

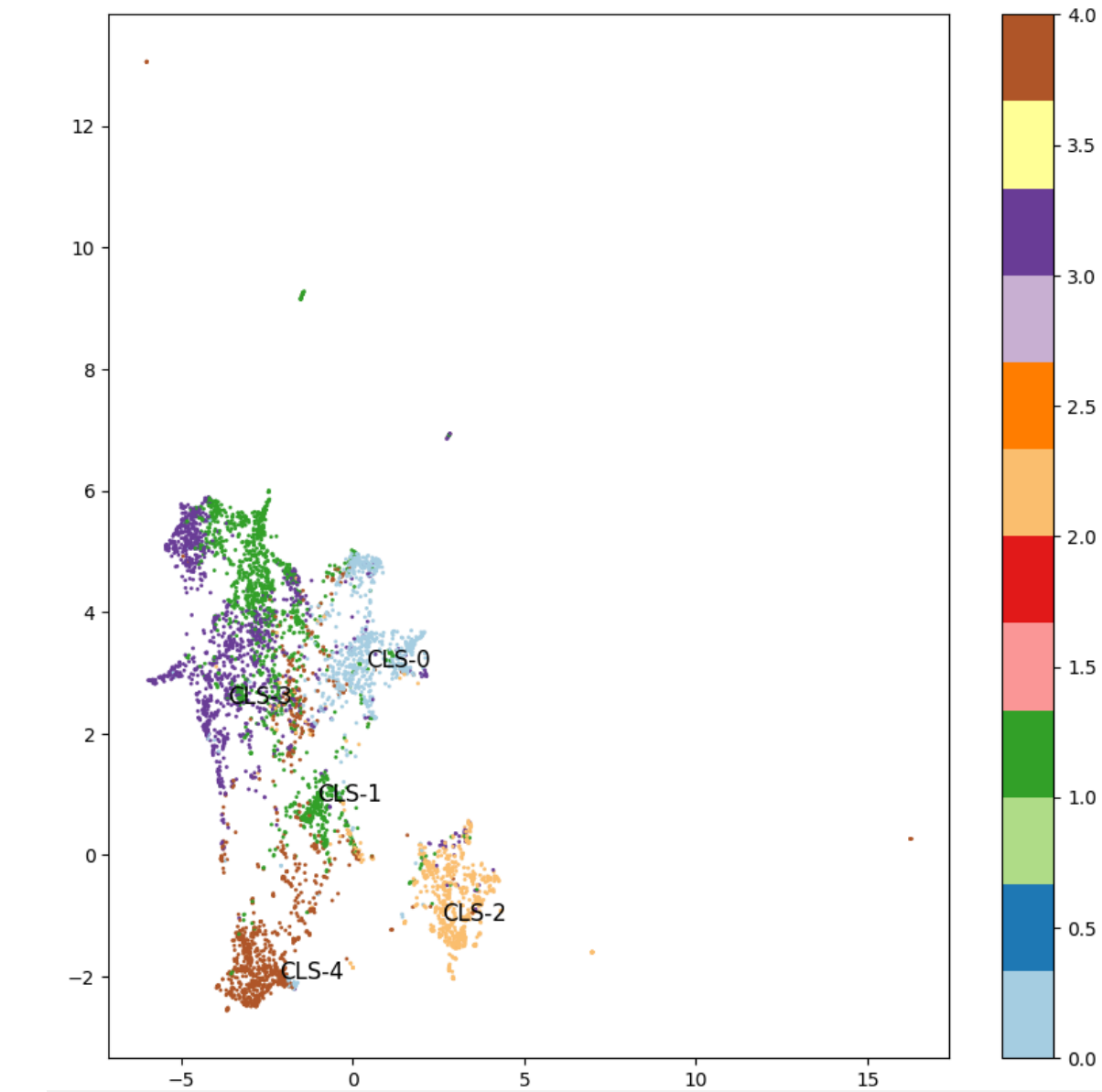
	count
1	1493
3	1378
4	1063
0	838
2	802

```
# !pip install umap-learn --quiet


distances = scipy.spatial.distance.cdist(kmeans.cluster_centers_,conversation_embeddings)
centers={}
print("Cluster", "Size", "Center-idx","Center-Example", sep="\t\t")
for i,d in enumerate(distances):
    ind = np.argsort(d, axis=0)[0]
    centers[i]=ind
    print(i,cls_dist[i], ind, df['process_text'][ind] ,sep="\t\t")
```

```
Cluster         Size            Center-idx              Center-Example
0               838             2942            My supervisor find 4 me one lor i thk his students. I havent ask her yet. Tell u aft i ask her.
1               1493            513             Lol ok your forgiven :)
2               802             389             4mths half price Orange line rental & latest camera phones 4 FREE. Had your phone 11mths ? Call MobilesDirect free on 080009387
3               1378            1881            Just seeing your missed call my dear brother. Do have a gr8 day.
4               1063            956             Sorry i now then c ur msg... Yar lor so poor thing... But only 4 one night... Tmr u'll have a brand new room 2 sleep in...
```

```
import matplotlib.pyplot as plt
import umap
X = umap.UMAP(n_components=2,min_dist=0.0).fit_transform(conversation_embeddings)
labels= kmeans.labels_
# print(labels)
fig, ax = plt.subplots(figsize=(10,10))
# print(X[:,0])\
plt.scatter(X[:,0], X[:,1], c=labels, s=1, cmap='Paired')
for c in centers:
    plt.text(X[centers[c],0], X[centers[c], 1],"CLS-"+ str(c), fontsize=12)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7c8ca789dfc0>
```



Text Classification using Albert model

Importing library

```
import numpy as np
import pandas as pd
import os
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import StratifiedKFold
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, Dataset
```

```
from tqdm import tqdm
import matplotlib.pyplot as plt
import transformers
import random
# import chardet
import warnings

warnings.simplefilter('ignore')



scaler = torch.cuda.amp.GradScaler()
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
device
```

```
device(type='cuda')
```

```
def random_seed(SEED):

    random.seed(SEED)
    os.environ['PYTHONHASHSEED'] = str(SEED)
    np.random.seed(SEED)
    torch.manual_seed(SEED)
    torch.cuda.manual_seed(SEED)
    torch.cuda.manual_seed_all(SEED)
    torch.backends.cudnn.deterministic = True

SEED = 508
random_seed(SEED)
```

Loading training file

```
data=pd.read_csv('/content/SMS_spam_detection_train.csv')
# data.columns=['text','lable']

display(data)
classes=sorted(data['label'].unique().tolist())

print(classes)
#
class_names=list('01')

N=list(range(len(class_names)))

normal_mapping=dict(zip(class_names,N))

reverse_mapping=dict(zip(N,class_names))
```

	sms	label
0	Go until jurong point, crazy.. Available only ...	0
1	Ok lar... Joking wif u oni...\n	0
2	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	U dun say so early hor... U c already then say...	0
4	Nah I don't think he goes to usf, he lives aro...	0
...	...	...
5569	This is the 2nd time we have tried 2 contact u...	1
5570	Will ü b going to esplanade fr home?\n	0
5571	Pity, * was in mood for that. So...any other s...	0
5572	The guy did some bitching but I acted like i'd...	0
5573	Rofl. Its true to its name\n	0

5574 rows × 2 columns

Train - Test split

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size=0.2, random_state=42)


#! pip install sentencepiece --quiet
```

Loading tokenizer model

```
tokenizer = transformers.AlbertTokenizer.from_pretrained("albert-base-v2")
```

```
tokenizer_config.json: 100%  25.0/25.0 [00:00<00:00, 1.99kB/s]
spiece.model: 100%  760k/760k [00:00<00:00, 4.54MB/s]
tokenizer.json: 100%  1.31M/1.31M [00:00<00:00, 13.6MB/s]
config.json: 100%  684/684 [00:00<00:00, 45.6kB/s]
```

Encoding sentence

```
test_s = train['sms'].iloc[0]
result1 = tokenizer.encode_plus(test_s)
tokenizer.decode(result1["input_ids"])
```

```
len(test_s.split(" "))
```

```
26
```

```
result2 = tokenizer.encode_plus(
    test_s,
    add_special_tokens = True,
    max_length = 8,
    pad_to_max_length = True,
    truncation = True
)
```

```
tokenizer.decode(result2["input_ids"])
```

```
'[CLS] free2day sexy at zaanga[SEP]'
```

```
max_sens = 8

train = train.sort_values('label').reset_index(drop=True)

train["kfold"] = train.index % 5

p_train = train[train["kfold"]!=0].reset_index(drop=True)
p_valid = train[train["kfold"]==0].reset_index(drop=True)

p_test=test.reset_index(drop=True)
```

Data loaders

```
class BERTDataSet(Dataset):
    def __init__(self,sentences,targets):
        self.sentences = sentences
        self.targets = targets

    def __len__(self):
        return len(self.sentences)

    def __getitem__(self,idx):
        sentence = self.sentences[idx]
        bert_sens = tokenizer.encode_plus(
                                sentence,
                                add_special_tokens = True,
                                max_length = max_sens,
                                pad_to_max_length = True,
                                return_attention_mask = True)

        ids = torch.tensor(bert_sens['input_ids'], dtype=torch.long)
        mask = torch.tensor(bert_sens['attention_mask'], dtype=torch.long)

        target = torch.tensor(self.targets[idx],dtype=torch.float)

        return {
                'ids': ids,
                'mask': mask,
                'targets': target
            }


train_dataset = BERTDataSet(p_train["sms"],p_train['label'])
valid_dataset = BERTDataSet(p_valid["sms"],p_valid['label'])
test_dataset = BERTDataSet(p_test["sms"],p_test['label'])

train_batch = 32
valid_batch = 32
test_batch = 32

train_dataloader = DataLoader(train_dataset,batch_size=train_batch,shuffle = True,num_workers=1,pin_memory=True)
valid_dataloader = DataLoader(valid_dataset,batch_size=valid_batch,shuffle = False,num_workers=1,pin_memory=True)
test_dataloader = DataLoader(test_dataset,batch_size=test_batch,shuffle = False,num_workers=1,pin_memory=True)
```

Loading transformer model

```
model = transformers.AlbertForSequenceClassification.from_pretrained("albert-base-v2", num_labels=1)
```

```
model.safetensors:  100%                                47.4M/47.4M  [00:00<00:00,  149MB/s]
Some weights of AlbertForSequenceClassification were not initialized from the model checkpoint at albert-base-v2 and are newly initialized: ['classifier.bias', 'classifier.wei
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
model.to(device)
model.train()
```

```
AlbertForSequenceClassification(
  (albert): AlbertModel(
    (embeddings): AlbertEmbeddings(
      (word_embeddings): Embedding(30000, 128, padding_idx=0)
      (position_embeddings): Embedding(512, 128)
      (token_type_embeddings): Embedding(2, 128)
      (LayerNorm): LayerNorm((128,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0, inplace=False)
    )
    (encoder): AlbertTransformer(
      (embedding_hidden_mapping_in): Linear(in_features=128, out_features=768, bias=True)
      (albert_layer_groups): ModuleList(
        (0): AlbertLayerGroup(
          (albert_layers): ModuleList(
            (0): AlbertLayer(
              (full_layer_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (attention): AlbertSdpaAttention(
                (query): Linear(in_features=768, out_features=768, bias=True)
```

```
                  (key): Linear(in_features=768, out_features=768, bias=True)
                  (value): Linear(in_features=768, out_features=768, bias=True)
                  (attention_dropout): Dropout(p=0, inplace=False)
                  (output_dropout): Dropout(p=0, inplace=False)
                  (dense): Linear(in_features=768, out_features=768, bias=True)
                  (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
                )
                (ffn): Linear(in_features=768, out_features=3072, bias=True)
                (ffn_output): Linear(in_features=3072, out_features=768, bias=True)
                (activation): NewGELUActivation()
                (dropout): Dropout(p=0, inplace=False)
              )
            )
          )
        )
      )
      (pooler): Linear(in_features=768, out_features=768, bias=True)
      (pooler_activation): Tanh()
    )
    (dropout): Dropout(p=0.1, inplace=False)
    (classifier): Linear(in_features=768, out_features=1, bias=True)
  )
```

```
# for a in train_dataloader:
#   print(a)
#   break
```

loss function

```
def loss_fn(output,target):
    return torch.sqrt(nn.MSELoss()(output,target))
```

Training function

```
def training(
    train_dataloader,
    model,
    optimizer,
    scheduler
):

    model.train()
    torch.backends.cudnn.benchmark = True
    allpreds = []
    alltargets = []

    for a in train_dataloader:

        losses = []
        optimizer.zero_grad()

        with torch.cuda.amp.autocast():

            ids = a["ids"].to(device,non_blocking=True)
            mask = a["mask"].to(device,non_blocking=True)

            output = model(ids,mask)
            output = output["logits"].squeeze(-1)
            target = a["targets"].to(device,non_blocking=True)
            loss = loss_fn(output,target)

            losses.append(loss.item())
            allpreds.append(output.detach().cpu().numpy())
            alltargets.append(target.detach().squeeze(-1).cpu().numpy())

        scaler.scale(loss).backward()
        scaler.step(optimizer)
        scaler.update()
        del loss

        scheduler.step()

    allpreds = np.concatenate(allpreds)
    alltargets = np.concatenate(alltargets)
    losses = np.mean(losses)
    train_rme_loss = np.sqrt(mean_squared_error(alltargets,allpreds))

    return losses,train_rme_loss
```

Validation function

```
def validating(valid_dataloader,model):

    model.eval()
    allpreds = []
    alltargets = []

    for a in valid_dataloader:
        losses = []
        with torch.no_grad():

            ids = a["ids"].to(device)
            mask = a["mask"].to(device)

            output = model(ids,mask)
            output = output["logits"].squeeze(-1)
            target = a["targets"].to(device)
            loss = loss_fn(output,target)
            losses.append(loss.item())
            allpreds.append(output.detach().cpu().numpy())
```

```
            alltargets.append(target.detach().squeeze(-1).cpu().numpy())

            del loss

    allpreds = np.concatenate(allpreds)
    alltargets = np.concatenate(alltargets)
    losses = np.mean(losses)
    valid_rme_loss = np.sqrt(mean_squared_error(alltargets,allpreds))

    return allpreds,losses,valid_rme_loss
```

Setting up optimizer

```
from transformers import AdamW
LR=2e-5
optimizer = AdamW(model.parameters(), LR,betas=(0.9, 0.999), weight_decay=1e-2)
```

Training process

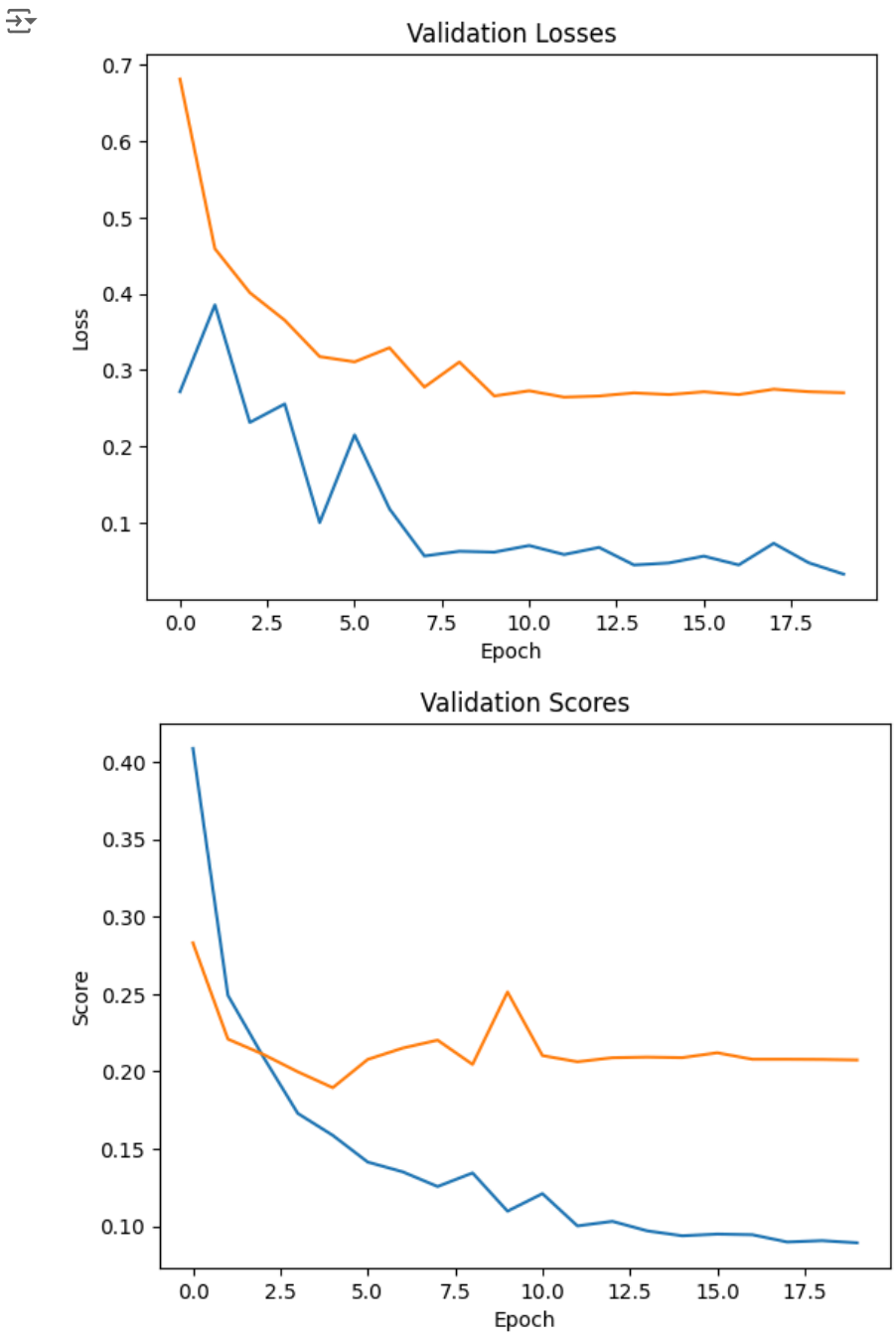
```
from transformers import get_linear_schedule_with_warmup
epochs = 20
#if debug:
#    epochs = 1
train_steps = int(len(p_train)/train_batch*epochs)
print(train_steps)
num_steps = int(train_steps*0.1)
scheduler = get_linear_schedule_with_warmup(optimizer, num_steps, train_steps)
```

```
2229
```

```
trainlosses = []
vallosses = []
bestscore = None
trainscores = []
validscores = []

for epoch in tqdm(range(epochs)):

    print("---------------" + str(epoch) + "start-------------")

    trainloss,trainscore = training(train_dataloader,model,optimizer,scheduler)
    trainlosses.append(trainloss)
    trainscores.append(trainscore)

    print("trainscore is " + str(trainscore))

    preds,validloss,valscore=validating(valid_dataloader,model)
    vallosses.append(validloss)
    validscores.append(valscore)

    print("valscore is " + str(valscore))

    if bestscore is None:
        bestscore = valscore

        print("Save first model")

        state = {
                        'state_dict': model.state_dict(),
                        'optimizer_dict': optimizer.state_dict(),
                        "bestscore":bestscore
                    }

        # torch.save(state, "model0.pth")

    elif bestscore > valscore:

        bestscore = valscore

        print("found better point")
        state = {
                        'state_dict': model.state_dict(),
                        'optimizer_dict': optimizer.state_dict(),
                        "bestscore":bestscore
                    }

        # torch.save(state, "model0.pth")

    else:
        pass
```

```
 70%|███████   | 14/20 [01:32<00:38,  6.38s/it]valscore is 0.20924402310780776
---------------14start-------------
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
trainscore is 0.09375087916438814
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
 75%|███████   | 15/20 [01:39<00:32,  6.46s/it]valscore is 0.20889819676776866
---------------15start-------------
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
trainscore is 0.09484380172149907
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
 80%|████████  | 16/20 [01:45<00:24,  6.24s/it]valscore is 0.21204073587449307
---------------16start-------------
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
trainscore is 0.09444767414043087
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
 85%|████████  | 17/20 [01:52<00:19,  6.47s/it]valscore is 0.2078543800087049
---------------17start-------------
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
trainscore is 0.0897495460053812
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
 90%|█████████ | 18/20 [01:58<00:12,  6.23s/it]valscore is 0.2078390645852772
---------------18start-------------
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
trainscore is 0.09062187501846904
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
 95%|█████████| 19/20 [02:05<00:06,  6.55s/it]valscore is 0.20773112561000398
---------------19start-------------
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
trainscore is 0.08919293572983909
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting
100%|██████████| 20/20 [02:11<00:00,  6.55s/it]valscore is 0.20738852460894472
```

```
# plt.scatter(p_valid['label'],preds, alpha=0.2)
# plt.title('Validation Prediction Result')
# plt.xlabel('Actual')
# plt.ylabel('Prediction')
# plt.show()

x = np.arange(epochs)
plt.title('Validation Losses')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.plot(x,trainlosses)
plt.plot(x,vallosses)
plt.show()

x = np.arange(epochs)
plt.title('Validation Scores')
plt.xlabel('Epoch')
plt.ylabel('Score')
plt.plot(x,trainscores)
plt.plot(x,validscores)
plt.show()
```




```
p_valid
```

	sms	label	kfold
0	Spoke with uncle john today. He strongly feels...	0	0
1	It wont b until 2.15 as trying 2 sort house ou...	0	0
2	Once free call me sir.\n	0	0
3	Come to mahal bus stop.. <DECIMAL>\n	0	0
4	Headin towards busetop\n	0	0
...	...	...	...
887	Do you want 750 anytime any network mins 150 t...	1	0
888	SMS. ac Blind Date 4U!: Rodds1 is 21/m from Ab...	1	0
889	You have 1 new voicemail. Please call 08719181...	1	0
890	Congratulations ur awarded 500 of CD vouchers ...	1	0
891	You have an important customer service announc...	1	0

Validation classification report

```
val_true = p_valid['label']
val_pred = []
for p in preds:
    val_pred+=[round(p,0)]

from sklearn.metrics import classification_report
print(classification_report(val_true,val_pred,target_names=class_names,digits=4))
```

```
              precision    recall  f1-score   support

           0     0.9577    0.9923    0.9747       775
           1     0.9326    0.7094    0.8058       117

    accuracy                         0.9552       892
   macro avg     0.9451    0.8508    0.8902       892
weighted avg     0.9544    0.9552    0.9525       892
```

Prediction on test dataset

```
def predicting(test_dataloader,model):

    model.to(device)
    model.eval()
    allpreds = []
    preds = []
    allvalloss=0

    with torch.no_grad():
        for a in test_dataloader:

            ids = a["ids"].to(device)
            mask = a["mask"].to(device)

            output = model(ids,mask)
            output = output["logits"].squeeze(-1)
            preds.append(output.cpu().numpy())

        preds = np.concatenate(preds)
        allpreds.append(preds)

    return allpreds


tpreds = predicting(test_dataloader, model)
```

```
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting t
```

```
# tpreds
```

Prediction classification report

```
test_true = p_test['label']
test_pred = []
for p in tpreds[0]:
    test_pred+=[round(p,0)]

from sklearn.metrics import classification_report
print(classification_report(test_true,test_pred,target_names=class_names,digits=4))
```

```
              precision    recall  f1-score   support

           0     0.9613    0.9885    0.9747       954
           1     0.9179    0.7640    0.8339       161

    accuracy                         0.9561      1115
   macro avg     0.9396    0.8762    0.9043      1115
weighted avg     0.9550    0.9561    0.9543      1115
```