Identify the Sentiments

Final Huanxin Sheng 15220202202189

This is the final project of

Artificial Intelligence and Machine Learning(2023Fall) taught by Prof.Zhonglei Wang at WISE, XMU

In [ ]:

```
!pip install wordninja
!pip install unidecode
!pip install ktrain
!pip install blurr
```

In [ ]:

```
import re
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import wordninja
import unidecode
import ktrain
```

1 Data

Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. Brands can use this data to measure the success of their products in an objective manner. In this challenge, you are provided with tweet data to predict sentiment on electronic products of netizens.

Contest/Data Source

1.1 Data Loading

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.moun
t("/content/drive", force_remount=True).
```

In [ ]:

```
from pathlib import Path
path = Path('/content/drive/MyDrive/人工智能与机器学习/Final-NLP')
```

1.2 Data Preprocessing

In [ ]:

```
def clean_tweet(text):

    # lower-case all characters
    text=text.lower()

    # remove twitter handles
    text= re.sub(r'@\S+', '',text)

    # remove urls
    text= re.sub(r'http\S+', '',text)
    text= re.sub(r'pic.\S+', '',text)

    # replace unidecode characters
    text=unidecode.unidecode(text)

    # regex only keeps characters
    text= re.sub(r"[^a-zA-Z+']", ' ',text)

    # keep words with length>1 only
    text=re.sub(r'\s+[a-zA-Z]\s+', ' ', text+' ')

    # split words like 'whatisthis' to 'what is this'
    def preprocess_wordninja(sentence):
        def split_words(x):
            x=wordninja.split(x)
            x= [word for word in x if len(word)>1]
            return x
        new_sentence=[ ' '.join(split_words(word)) for word in sentence.split() ]
        return ' '.join(new_sentence)

    text=preprocess_wordninja(text)

    # regex removes repeated spaces, strip removes leading and trailing spaces
    text= re.sub("\s[\s]+", " ",text).strip()

    return text
```

In [ ]:

```
train_df = pd.read_csv(path/'train_2kmZucJ.csv')
train_df = train_df.rename(columns={'tweet':'text'})
train_df['text']=train_df['text'].apply(lambda x: clean_tweet(x))
train_df.head()
```

Out[ ]:

	id	label	text
0	1	0	fingerprint pregnancy test android apps beauti...
1	2	0	finally trans paran silicon case thanks to my ...
2	3	0	we love this would you go talk make memories u...
3	4	0	i'm wired know i'm george was made that way ip...
4	5	1	what amazing service apple won't even talk to ...

In [ ]:

```
# Count number of Positive->0 and Negative->1 in training data
train_df['label'].value_counts()
```

Out[ ]:

```
0    5894
1    2026
Name: label, dtype: int64
```

In [ ]:

```
test_df = pd.read_csv(path/'test_oJQbWVk.csv')
test_df = test_df.rename(columns={'tweet':'text'})
test_df['text']=test_df['text'].apply(lambda x: clean_tweet(x))
test_df.head()
```

Out[ ]:

	id	text
0	7921	hate the new iphone upgrade won't let me downl...
1	7922	currently shitting my fucking pants apple imac...

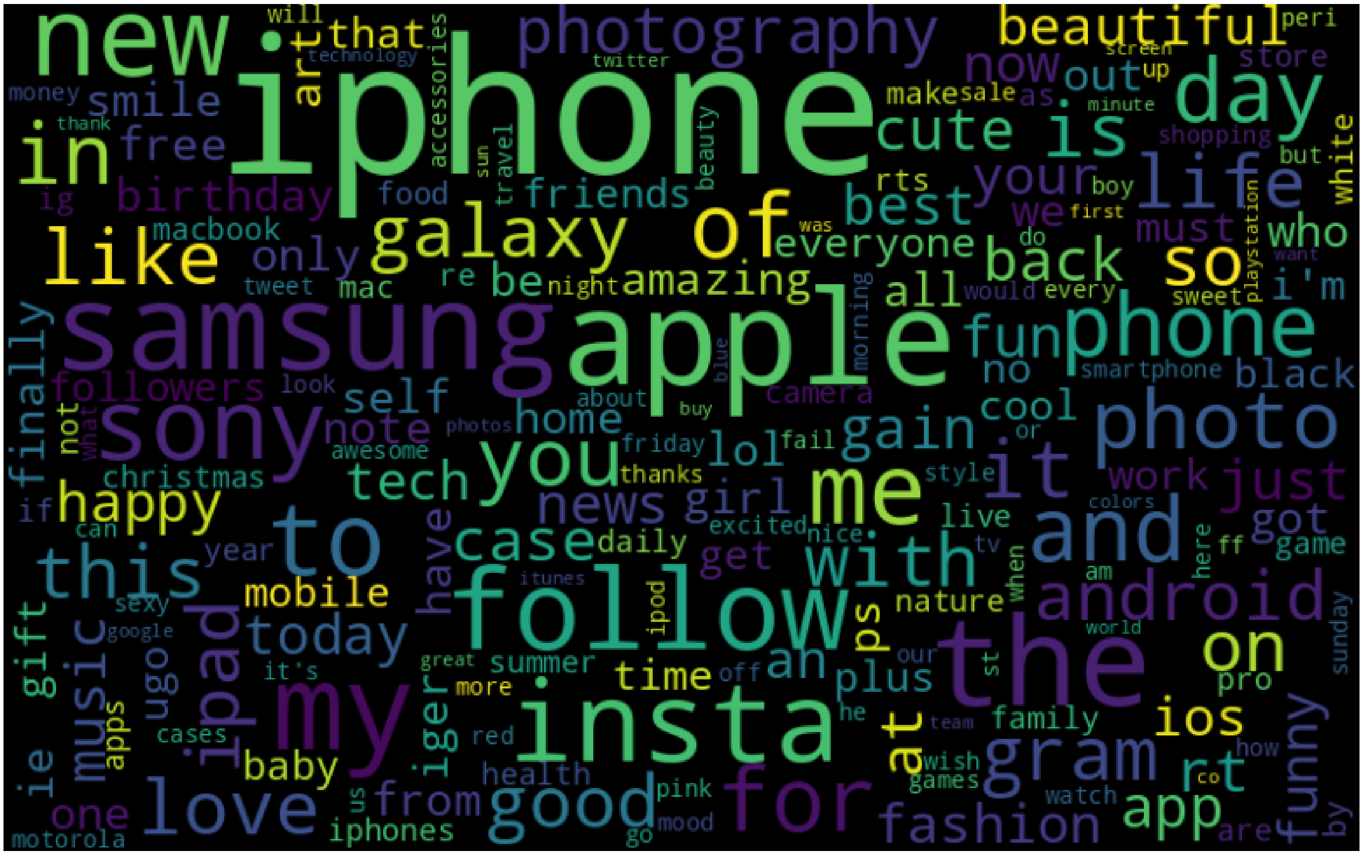
2	7923	i'd like to puts some cd roms on my ipad is th...
3	7924	my ipod is officially dead lost all my and vid...
4	7925	been fighting itunes all night only want the m...

1.3 Word Cloud Depiction

In [ ]:

```
positive= train_df[train_df['label']==0]
all_words = ' '.join([text for text in positive['text']])

#Counting Frequency
from collections import Counter
word_counts = Counter(all_words.split())
sorted_word_counts = dict(sorted(word_counts.items(), key=lambda x: x[1], reverse=True))

#Word Cloud Depiction
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=110, colorma

plt.figure(figsize=(10,8))
plt.imshow(wordcloud, interpolation="bilinear")
plt.title('Word Cloud of Positive Tweets')
plt.axis('off')
plt.show()
```

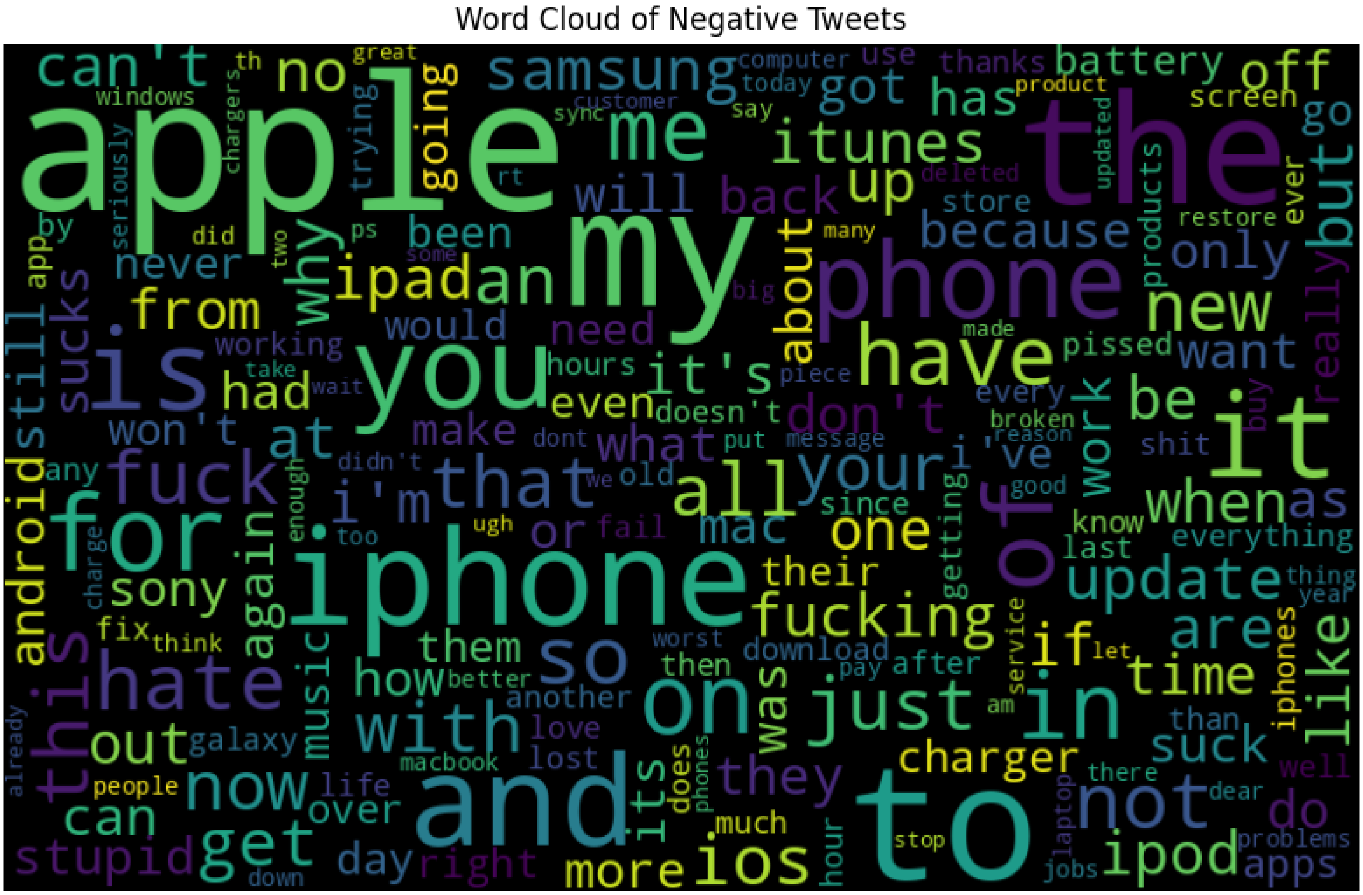
Word Cloud of Positive Tweets



In [ ]:

```
negative= train_df[train_df['label']==1]
all_words = ' '.join([text for text in negative['text']])

#Counting Frequency
from collections import Counter
word_counts = Counter(all_words.split())
sorted_word_counts = dict(sorted(word_counts.items(), key=lambda x: x[1], reverse=True))
```

```
#Word Cloud Depiction
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=110, colorma

plt.figure(figsize=(10,8))
plt.imshow(wordcloud, interpolation="bilinear")
plt.title('Word Cloud of Negative Tweets')
plt.axis('off')
plt.show()
```



Word Cloud of Negative Tweets

2 Model

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(train_df.drop(['label'], axis=1), trai
```

In [ ]:
```
X_train
```

Out[ ]:

	id	text
4252	4253	cool car wash idea the island bank holiday mon...
4428	4429	photo th birthday to the sony walkman at nobod...
7374	7375	ipad are the biggest pile of fucking on the pl...
1410	1411	yearbook hmmm mm insta gram insta good togethe...
7896	7897	so pissed macbook crashes apple company does n...
...	...	...
5226	5227	shana to va jewish new year everyone may your ...
5390	5391	i'm so sick of buying new cell phone chargers ...
860	861	it want it have it download the free iphone ap...

7603	7604	photo nikos iphone beach holiday iphone black ...
7270	7271	just got an iphone he he iphone apple new fina...

6336 rows × 2 columns

In [ ]:

```
y_train
```

Out[ ]:

```
4252    0
4428    0
7374    1
1410    0
7896    1
       ..
5226    0
5390    1
860     0
7603    0
7270    1
Name: label, Length: 6336, dtype: int64
```

In [ ]:

```
text_column = 'text'
if text_column in X_train.columns:
    train_list = X_train[text_column].astype(str).tolist()
else:
    print(f"Error: '{text_column}' column not found in the DataFrame.")

if text_column in X_val.columns:
    val_list = X_val[text_column].astype(str).tolist()
else:
    print(f"Error: '{text_column}' column not found in the DataFrame.")
```

2.1 Model 1: AWD-LSTM

Stephen Merity, Nitish Shirish Keskar, Richard Socher: Regularizing and Optimizing LSTM Language Models. ICLR (Poster) 2018

In [ ]:

```
from fastai.text.all import *
```

In [ ]:

```
## Creating a Text Data loader for language model
df_all = pd.concat([train_df.drop(['label'], axis=1),test_df])
TDL_Train = TextDataLoaders.from_df(df_all, text_col=1, is_lm=True)
TDL_Train.show_batch(max_n=5)
```

	text	text_
0	xxbos live out loud lol live out loud self ie smile sony music headphones xxbos follow on insta gram sup surf fun capetown funny sexy me samsung xxbos see you later tonight xxunk club france edm edm fam iphone dance edc lv pl ur xxbos facetime with my sister and niece family xxwrep 3 iphone xxbos anybody listened to the new yet took me all night to xxunk itunes to let me	live out loud lol live out loud self ie smile sony music headphones xxbos follow on insta gram sup surf fun capetown funny sexy me samsung xxbos see you later tonight xxunk club france edm edm fam iphone dance edc lv pl ur xxbos facetime with my sister and niece family xxwrep 3 iphone xxbos anybody listened to the new yet took me all night to xxunk itunes to let me have
1	'm media and attention xxunk xxbos my first new phone since quite xxunk it xxunk far and really love having droid phone again motorola xxbos happy birthday mother xxunk xxunk xxunk an birthday mom apple iphone plus xxbos dell computers literally cause more stress then anything else in the entire world hate you where my mac apple xxbos back	media and attention xxunk xxbos my first new phone since quite xxunk it xxunk far and really love having droid phone again motorola xxbos happy birthday mother xxunk xxunk xxunk an birthday mom apple iphone plus xxbos dell computers literally cause more stress then anything else in the entire world hate you where my mac apple xxbos back

	to my city life goodnight music art apple iphone only xxunk xxunk hair	to my city life goodnight music art apple iphone only xxunk xxunk hair jesus
2	apple xxunk strawberry oreo joy dinner xxunk xxbos iphone app now health fitness htc one samsung galaxy tab iphone case can iphone xxbos xxunk of electronic heads electronic dance dancing celebration people samsung xxbos for my xxunk xxunk friend high school xxunk samsung galaxy friendship happiness score xxbos who 's feeling the for xxunk xxunk in our classic wood case silver and gold we xxunk iphone iphone case xxbos technology actually hates	xxunk strawberry oreo joy dinner xxunk xxbos iphone app now health fitness htc one samsung galaxy tab iphone case can iphone xxbos xxunk of electronic heads electronic dance dancing celebration people samsung xxbos for my xxunk xxunk friend high school xxunk samsung galaxy friendship happiness score xxbos who 's feeling the for xxunk xxunk in our classic wood case silver and gold we xxunk iphone iphone case xxbos technology actually hates me
3	your crap ever rant xxbos post xxunk my samsung runs smooth at the moment have loaded my device with all memory xxunk apps and it does n't xxunk xxbos morning self ie me model iger iphone ig daily in st ago ig addict insta good hot insta mood xxbos what do you mean error restoring your ipod xxunk losing all my old music not on this laptop xxunk xxbos finally my ipad	crap ever rant xxbos post xxunk my samsung runs smooth at the moment have loaded my device with all memory xxunk apps and it does n't xxunk xxbos morning self ie me model iger iphone ig daily in st ago ig addict insta good hot insta mood xxbos what do you mean error restoring your ipod xxunk losing all my old music not on this laptop xxunk xxbos finally my ipad mini
4	beauty beautiful sexy fotos focus canon nikon sony faith follow xxbos happy days day diy cord protector thanks cute samsung headset insta mag android xxbos every day you save my life friday self ie blonde iphone smile xxbos facetime with these fools tonight brothers brother iphone facetime family technology xxbos the kid in the apple commercial that asks what computer is really pisses me off xxbos your ipod has xxunk fully been	beautiful sexy fotos focus canon nikon sony faith follow xxbos happy days day diy cord protector thanks cute samsung headset insta mag android xxbos every day you save my life friday self ie blonde iphone smile xxbos facetime with these fools tonight brothers brother iphone facetime family technology xxbos the kid in the apple commercial that asks what computer is really pisses me off xxbos your ipod has xxunk fully been jailbroken

In [ ]:

```
## Initiating the language model
AWD_LSTM_learn = language_model_learner(
    TDL_Train,
    AWD_LSTM,
    metrics=[accuracy, Perplexity()],
    path=path,
    wd=0.1).to_fp16()
```

```
100.00% [105070592/105067061 00:03<00:00]
```

In [ ]:

```
## Training last layer
AWD_LSTM_learn.fit_one_cycle(5, 1e-2)
```

epoch	train_loss	valid_loss	accuracy	perplexity	time
0	7.376301	5.999442	0.098768	403.203796	00:18
1	6.358181	5.366263	0.164037	214.061508	00:11
2	5.803487	5.170400	0.186619	175.985153	00:09
3	5.469845	5.103769	0.192560	164.641327	00:08
4	5.264073	5.092362	0.194415	162.773941	00:09

In [ ]:

```
## Training all layers
AWD_LSTM_learn.unfreeze()
AWD_LSTM_learn.fit_one_cycle(10, 1e-3)
```

epoch	train_loss	valid_loss	accuracy	perplexity	time
0	4.970094	4.995770	0.207584	147.786758	00:07

1	4.868682	4.886017	0.224630	132.425049	00:09
2	4.744607	4.798333	0.230492	121.307991	00:07
3	4.617784	4.726048	0.245522	112.848648	00:09
4	4.495970	4.701935	0.249937	110.160103	00:08
5	4.382375	4.677907	0.254134	107.544746	00:09
6	4.278681	4.670915	0.257072	106.795433	00:10
7	4.210018	4.670059	0.260264	106.704063	00:08
8	4.139768	4.671574	0.260523	106.865829	00:10
9	4.103185	4.672992	0.260170	107.017487	00:09

In [ ]:

```
## Saving encoder
AWD_LSTM_learn.save_encoder('AWD_LSTM_learn')
```

In [ ]:

```
## Creating a dataloader
TDL_Classify = TextDataLoaders.from_df(
    pd.concat([X_train, y_train], axis=1),
    valid_pct=0.2,
    seed=42,
    text_col=1,
    label_col=2,
    text_vocab=TDL_Train.vocab)
```

In [ ]:

```
## Checking if everything is working fine
TDL_Classify.show_batch(max_n=5)
```

	text	category
0	xxbos you know what apple it sucks that you ca n't update the older ipad to the newer ios because there are so many apps i 'd love to download but ca n't because they 're not compatible and in for king out xxup ps for new one when this one is still good apple ios	1
1	xxbos just like santa he is always watching dog dogs pet do glover puppy love pets tag ram adorable animal pets photo of the day insta good dogs dogs of insta gram doggy animals pets agram insta gram dogs pets of insta gram xxunk iphone mob it og apple ip ho	0
2	xxbos so week or so ago my samsung note updated and it 's been so annoying ever since they took out google talk to text and replaced it with their bixby finally downloaded the google keyboard finally google 's back no more editing it gets what i 'm saying right	0
3	xxbos it looks like andrew xxunk is going to xxunk the xxunk the big house an apple for xxunk the xxunk the xxunk big bob xxunk xxunk get ready for some hard time douche bag xxunk line xxunk fucked in the xxunk bitch lock andrew xxunk up loser as	0
4	xxbos love the way my stream is coming along head on over to twitch hit that follow button and turn on xxunk to see when go live twitch streamer ps sony gamer girl gamer chick twitch girl just dance over watch cod ww fps squad fun games xxunk	0

In [ ]:

```
## Defining our text classifier
AWD_LSTM_Classifier = text_classifier_learner(TDL_Classify , AWD_LSTM, drop_mult=1, metr
AWD_LSTM_Classifier.path = path
```

In [ ]:

```
## Loading Language model encoder weights trained in previous section
AWD_LSTM_Classifier = AWD_LSTM_Classifier.load_encoder(file='AWD_LSTM_learn')
```

In [ ]:

```
## Finetuning last layer
AWD_LSTM_Classifier.fit_one_cycle(5, slice(1e-4, 1e-2))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.498467	0.385782	0.876875	00:09
1	0.391104	0.258600	0.887135	00:10
2	0.352070	0.249460	0.891871	00:10
3	0.335830	0.248982	0.891871	00:07
4	0.321681	0.255494	0.891081	00:10

In [ ]:
```
## Finetuning last 2 layers and progressive learning rate
AWD_LSTM_Classifier.freeze_to(-2)
AWD_LSTM_Classifier.fit_one_cycle(5, slice(1e-4,1e-2))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.335544	0.260918	0.880821	00:08
1	0.322710	0.263299	0.886346	00:10
2	0.322468	0.257637	0.876875	00:09
3	0.300331	0.249230	0.888713	00:08
4	0.281105	0.250342	0.886346	00:09

In [ ]:
```
## Finetuning last 3 layers and progressive learning rate
AWD_LSTM_Classifier.freeze_to(-3)
AWD_LSTM_Classifier.fit_one_cycle(5, slice(1e-4,1e-2))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.281030	0.266678	0.881610	00:07
1	0.293262	0.247695	0.898185	00:10
2	0.275282	0.243719	0.895028	00:07
3	0.249205	0.248376	0.895817	00:10
4	0.238582	0.258991	0.891081	00:08

In [ ]:
```
## Training entire model
AWD_LSTM_Classifier.unfreeze()
AWD_LSTM_Classifier.fit_one_cycle(10, slice(1e-5,1e-3))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.232050	0.257192	0.893449	00:09
1	0.226816	0.255575	0.893449	00:10
2	0.226133	0.259606	0.893449	00:07
3	0.220889	0.257704	0.894238	00:10
4	0.218374	0.250843	0.898974	00:08
5	0.211722	0.255897	0.897395	00:10
6	0.209109	0.264999	0.902131	00:08
7	0.206350	0.260082	0.896606	00:09

8	0.209173	0.258191	0.902131	00:09
9	0.201087	0.262240	0.896606	00:08

In [ ]:

```
## Save model weights
AWD_LSTM_Classifier.save('AWD_LSTM_Classifier')
```

Out[ ]:

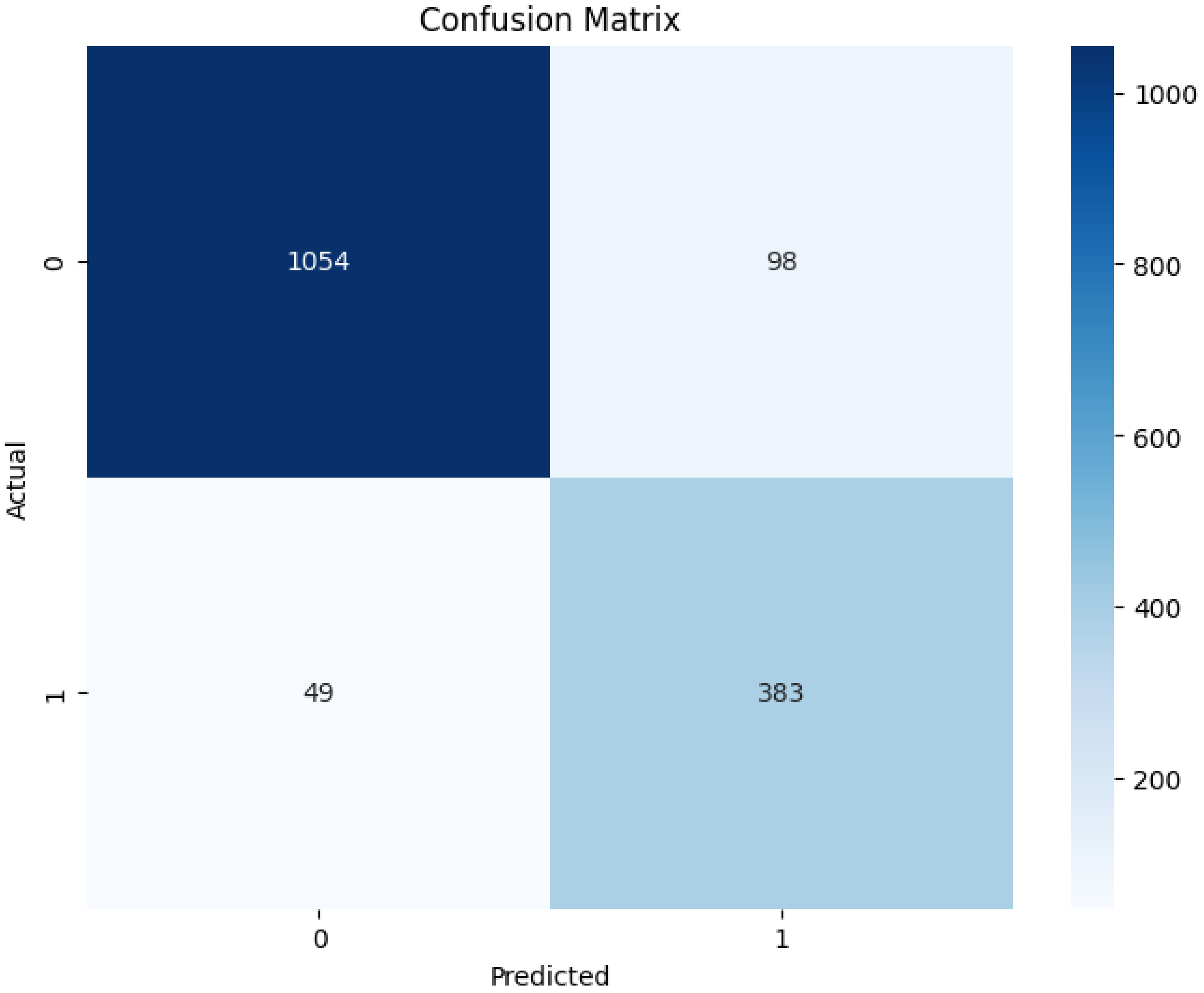
```
Path('/content/drive/MyDrive/人工智能与机器学习/Final-NLP/models/AWD_LSTM_Classifier.pth')
```

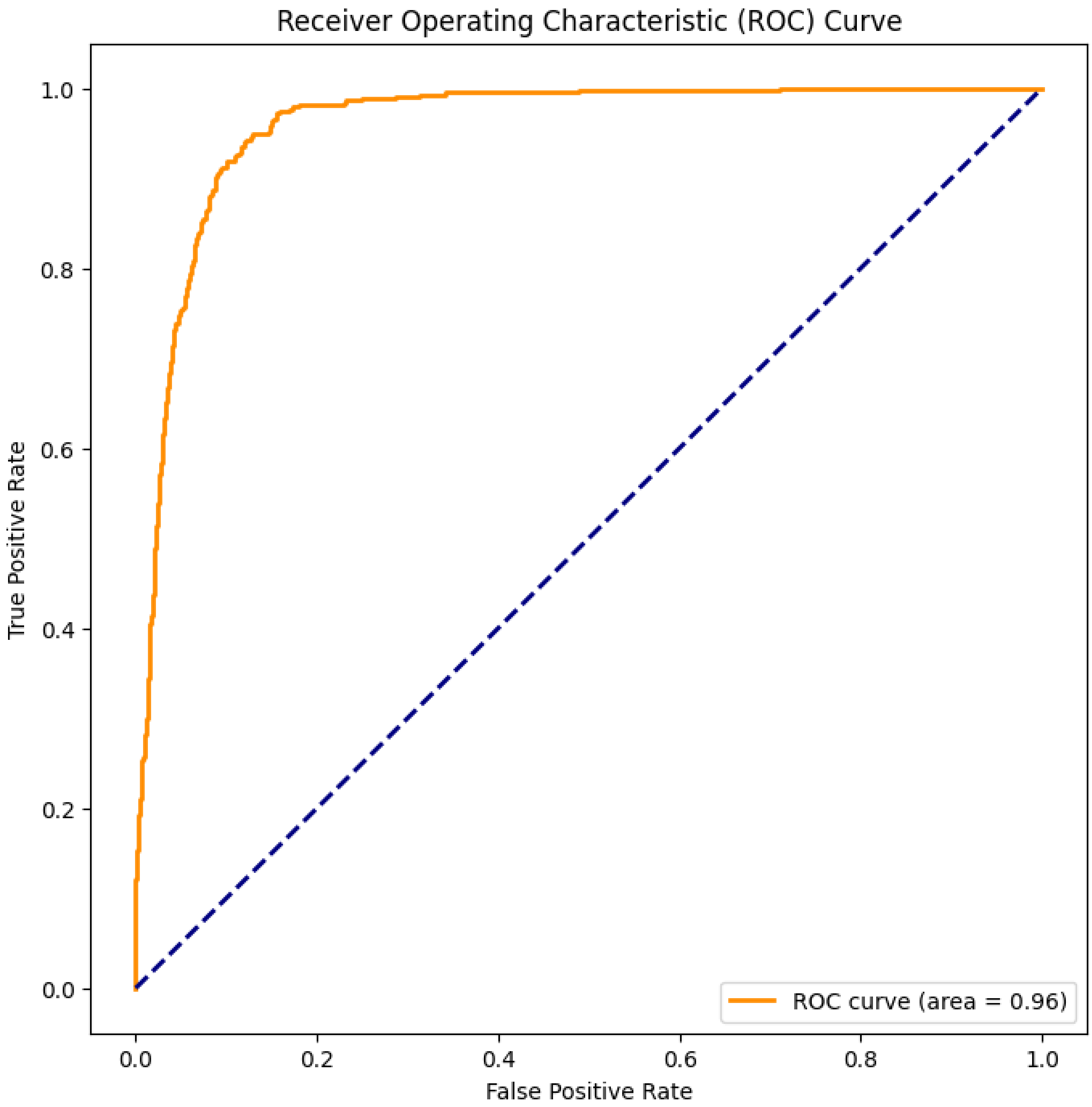
In [ ]:

```
## Getting testing data and doing predictions
test_dl = AWD_LSTM_Classifier.dls.test_dl(X_val)
preds, _ = AWD_LSTM_Classifier.get_preds(dl=test_dl)
ppp1 = preds[:, 1].numpy()#predicted_positive_probabilities
y_val = np.array(y_val)
```

In [ ]:

```
from sklearn.metrics import roc_curve, auc

from sklearn.metrics import confusion_matrix, roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt


# 计算并绘制混淆矩阵
conf_matrix = confusion_matrix(y_val, (ppp1 > 0.5).astype(int))
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], ytic
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

fpr, tpr, thresholds = roc_curve(y_val,ppp1)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 8))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = {:.2f})'.format(ro
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```


Confusion Matrix
Actual
Predicted
0
1
0
1
1054
98
49
383
1000
800
600
400
200



2.2 Model 2: Bert

In [ ]:

```
from transformers import BertTokenizer, BertForSequenceClassification

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
Bert_Based = BertForSequenceClassification.from_pretrained('bert-base-uncased')
```

In [ ]:

```
inputs = tokenizer(train_list, return_tensors="pt", truncation=True, padding=True)
labels = torch.tensor(y_train.tolist(), dtype=torch.float32)
```

In [ ]:

```
import torch
from torch.utils.data import DataLoader, TensorDataset
from tqdm import tqdm

# 将模型和数据移到GPU上
Bert_Based.to('cuda')
inputs_cuda = {key: value.to('cuda') for key, value in inputs.items()}
labels_tensor_cuda = labels.to('cuda')
```

```
dataset = TensorDataset(inputs_cuda['input_ids'], inputs_cuda['attention_mask'], labels_
dataloader = DataLoader(dataset, batch_size=128, shuffle=True)

optimizer = torch.optim.AdamW(Bert_Based.parameters(), lr=1e-5)
criterion = torch.nn.CrossEntropyLoss()

# 记录训练过程中的损失和准确率
train_loss_history = []
train_accuracy_history = []
```

In [ ]:

```
for epoch in range(5):
    total_loss = 0.0
    total_correct_predictions = 0
    total_samples = 0

    progress_bar = tqdm(enumerate(dataloader), total=len(dataloader), desc=f'Epoch {epoc

    for batch_idx, batch in progress_bar:
        optimizer.zero_grad()
        outputs = Bert_Based(input_ids=batch[0], attention_mask=batch[1])
        logits = outputs.logits
        loss = criterion(logits, batch[2].long())

        loss.backward()
        optimizer.step()

        total_loss += loss.item()

        _, predicted_labels = torch.max(logits, 1)
        total_correct_predictions += (predicted_labels == batch[2]).sum().item()
        total_samples += batch[2].size(0)

        progress_bar.set_postfix({'Loss': loss.item(), 'Accuracy': total_correct_predict

    epoch_loss = total_loss / len(dataloader)
    epoch_accuracy = total_correct_predictions / total_samples

    train_loss_history.append(epoch_loss)
    train_accuracy_history.append(epoch_accuracy)

    print(f'Epoch {epoch + 1}/{3}, Average Loss: {epoch_loss:.4f}, Average Accuracy: {ep
```

```
Epoch 1/5: 100%|██████████| 50/50 [00:18<00:00,  2.68it/s, Loss=0.245, Accuracy=0.798]
Epoch 1/3, Average Loss: 0.4215, Average Accuracy: 0.7981
Epoch 2/5: 100%|██████████| 50/50 [00:18<00:00,  2.67it/s, Loss=0.22, Accuracy=0.919]
Epoch 2/3, Average Loss: 0.2122, Average Accuracy: 0.9192
Epoch 3/5: 100%|██████████| 50/50 [00:19<00:00,  2.61it/s, Loss=0.0807, Accuracy=0.942]
Epoch 3/3, Average Loss: 0.1584, Average Accuracy: 0.9421
Epoch 4/5: 100%|██████████| 50/50 [00:18<00:00,  2.67it/s, Loss=0.109, Accuracy=0.96]
Epoch 4/3, Average Loss: 0.1215, Average Accuracy: 0.9601
Epoch 5/5: 100%|██████████| 50/50 [00:18<00:00,  2.67it/s, Loss=0.0686, Accuracy=0.974]
Epoch 5/3, Average Loss: 0.0889, Average Accuracy: 0.9740
```

In [ ]:

```
# 预测验证集
# 数据准备
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
val_tokenized = tokenizer(val_list, padding=True, truncation=True, return_tensors='pt')
val_list_cuda = {key: value.to('cuda') for key, value in val_tokenized.items()}
labels_val_tensor_cuda = torch.tensor(y_val, dtype=torch.long).to('cuda')

# 开始验证
Bert_Based.eval()
```

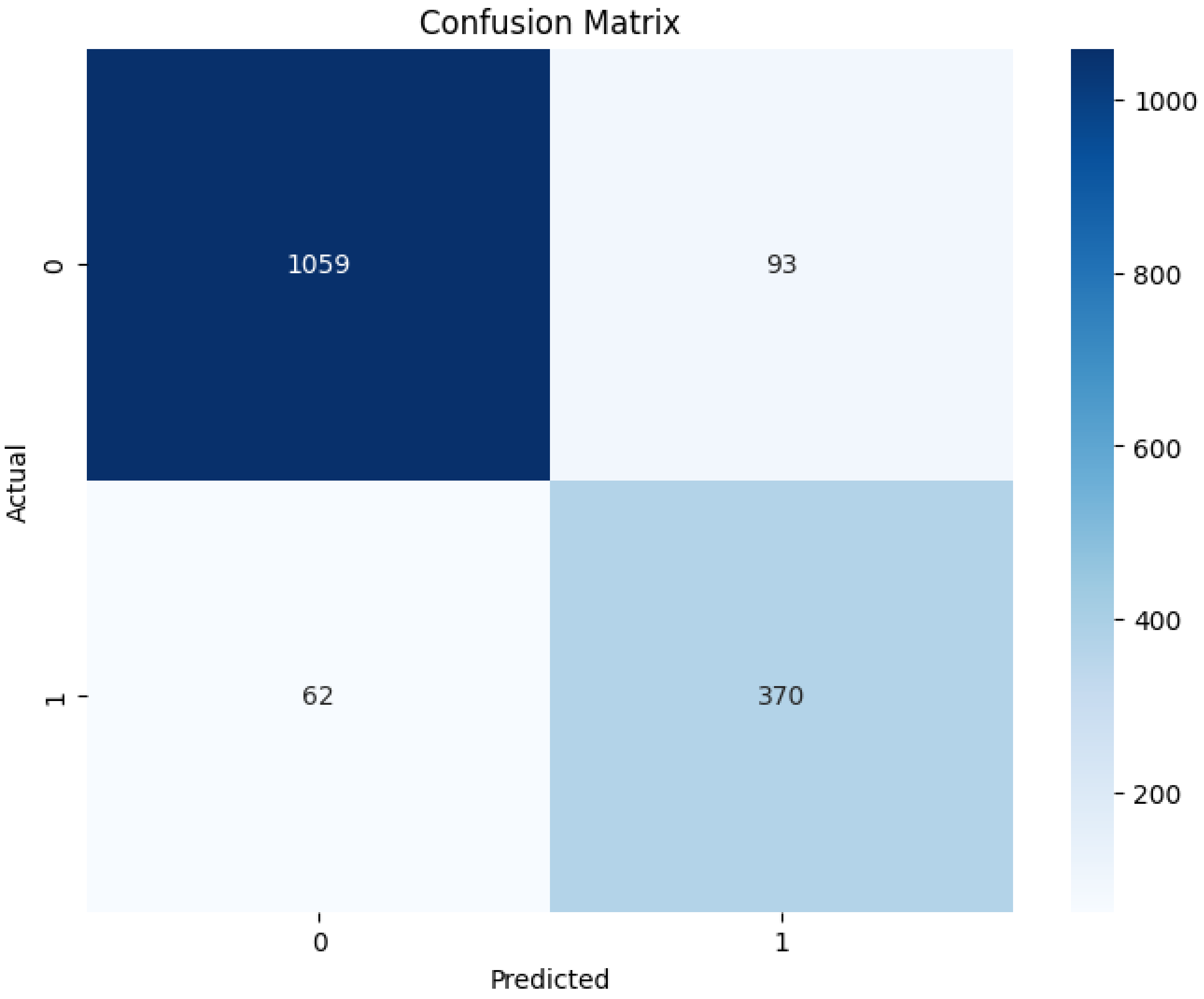
```
with torch.no_grad():
    outputs_val = Bert_Based(input_ids=val_list_cuda['input_ids'], attention_mask=val_li
    logits_val = outputs_val.logits
ppp2 = torch.softmax(logits_val, dim=1)[:, 1].cpu().numpy()
```

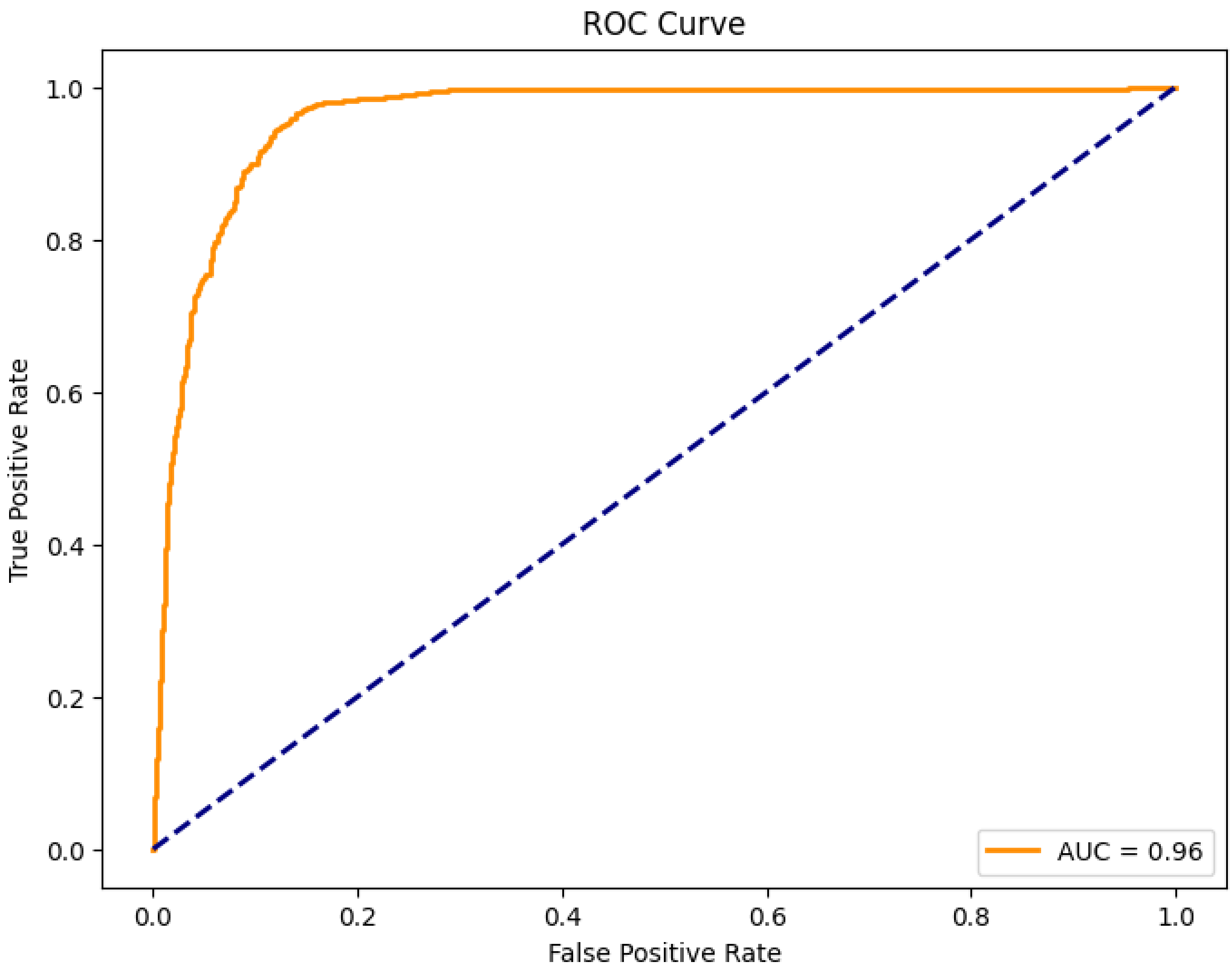
In [ ]:

```
# 保存模型的状态字典
model_state_dict = Bert_Based.state_dict()

# 保存模型参数
torch.save(model_state_dict, 'bert_based_model.pth')
```

In [ ]:

```
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt


# 计算并绘制混淆矩阵
conf_matrix = confusion_matrix(y_val, (ppp2 > 0.5).astype(int))
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], ytic
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# 计算并绘制ROC曲线
fpr, tpr, thresholds = roc_curve(y_val, ppp2)
roc_auc = roc_auc_score(y_val, ppp2)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```


Confusion Matrix
Actual
0
1
1059
93
62
370
0
1
Predicted
1000
800
600
400
200



2.3 Model 3: Roberta

In [ ]:

```
from transformers import RobertaTokenizer, RobertaForSequenceClassification
from torch.utils.data import DataLoader, TensorDataset
import torch
```

In [ ]:

```
# 加载 Roberta tokenizer 和模型
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
Roberta_Based = RobertaForSequenceClassification.from_pretrained('roberta-base', num_lab
```

In [ ]:

```
inputs = tokenizer(train_list, return_tensors="pt", truncation=True, padding=True)
labels = torch.tensor(y_train.tolist(), dtype=torch.float32)
```

In [ ]:

```
import torch
from torch.utils.data import DataLoader, TensorDataset
from tqdm import tqdm

# 将模型和数据移到GPU上
Roberta_Based.to('cuda')
inputs_cuda = {key: value.to('cuda') for key, value in inputs.items()}
labels_tensor_cuda = labels.to('cuda')

dataset = TensorDataset(inputs_cuda['input_ids'], inputs_cuda['attention_mask'], labels_
dataloader = DataLoader(dataset, batch_size=256, shuffle=True)

optimizer = torch.optim.AdamW(Roberta_Based.parameters(), lr=1e-5)
criterion = torch.nn.CrossEntropyLoss()
```

```
# 记录训练过程中的损失和准确率
train_loss_history = []
train_accuracy_history = []
```

In [ ]:

```
for epoch in range(5):
    total_loss = 0.0
    total_correct_predictions = 0
    total_samples = 0

    progress_bar = tqdm(enumerate(dataloader), total=len(dataloader), desc=f'Epoch {epoc

    for batch_idx, batch in progress_bar:
        optimizer.zero_grad()
        outputs = Roberta_Based(input_ids=batch[0], attention_mask=batch[1])
        logits = outputs.logits
        loss = criterion(logits, batch[2].long())

        loss.backward()
        optimizer.step()

        total_loss += loss.item()

        _, predicted_labels = torch.max(logits, 1)
        total_correct_predictions += (predicted_labels == batch[2]).sum().item()
        total_samples += batch[2].size(0)

        progress_bar.set_postfix({'Loss': loss.item(), 'Accuracy': total_correct_predict

    epoch_loss = total_loss / len(dataloader)
    epoch_accuracy = total_correct_predictions / total_samples

    train_loss_history.append(epoch_loss)
    train_accuracy_history.append(epoch_accuracy)

    print(f'Epoch {epoch + 1}/5, Average Loss: {epoch_loss:.4f}, Average Accuracy: {epoc
```

```
Epoch 1/5: 100%|██████████| 25/25 [00:19<00:00,  1.30it/s, Loss=0.401, Accuracy=0.726]
Epoch 1/5, Average Loss: 0.5107, Average Accuracy: 0.7259
Epoch 2/5: 100%|██████████| 25/25 [00:19<00:00,  1.30it/s, Loss=0.24, Accuracy=0.823]
Epoch 2/5, Average Loss: 0.2927, Average Accuracy: 0.8232
Epoch 3/5: 100%|██████████| 25/25 [00:19<00:00,  1.30it/s, Loss=0.18, Accuracy=0.917]
Epoch 3/5, Average Loss: 0.2010, Average Accuracy: 0.9170
Epoch 4/5: 100%|██████████| 25/25 [00:19<00:00,  1.30it/s, Loss=0.12, Accuracy=0.935]
Epoch 4/5, Average Loss: 0.1616, Average Accuracy: 0.9350
Epoch 5/5: 100%|██████████| 25/25 [00:19<00:00,  1.30it/s, Loss=0.137, Accuracy=0.948]
Epoch 5/5, Average Loss: 0.1381, Average Accuracy: 0.9481
```

In [ ]:

```
# 预测验证集
# 数据准备
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')  # 使用Roberta的tokenizer
val_tokenized = tokenizer(val_list, padding=True, truncation=True, return_tensors='pt')
val_list_cuda = {key: value.to('cuda') for key, value in val_tokenized.items()}
labels_val_tensor_cuda = torch.tensor(y_val, dtype=torch.long).to('cuda')

# 开始验证
Roberta_Based.eval()
with torch.no_grad():
    outputs_val = Roberta_Based(input_ids=val_list_cuda['input_ids'], attention_mask=val
    logits_val = outputs_val.logits
ppp3 = torch.softmax(logits_val, dim=1)[:, 1].cpu().numpy()
```

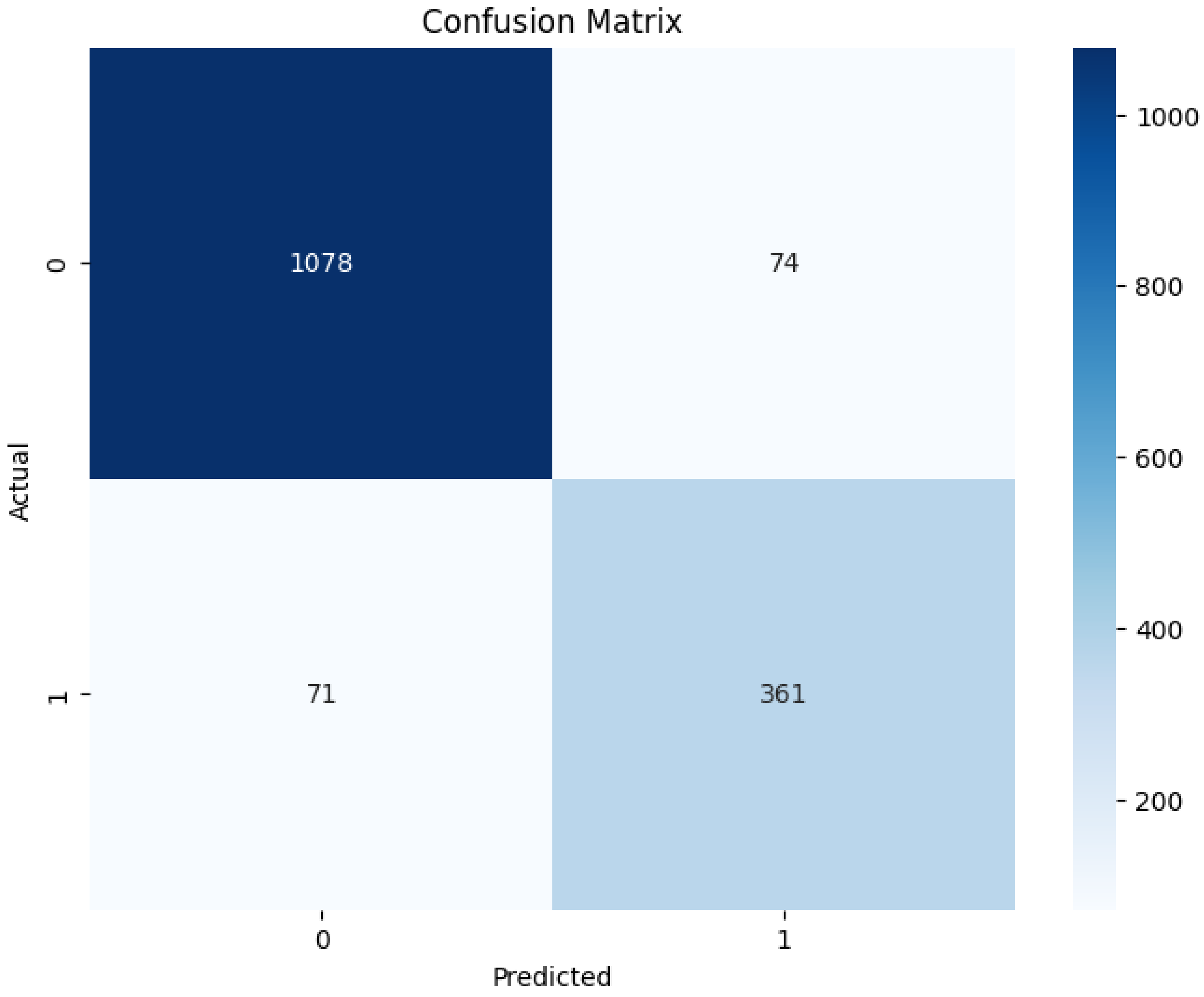
In [ ]:

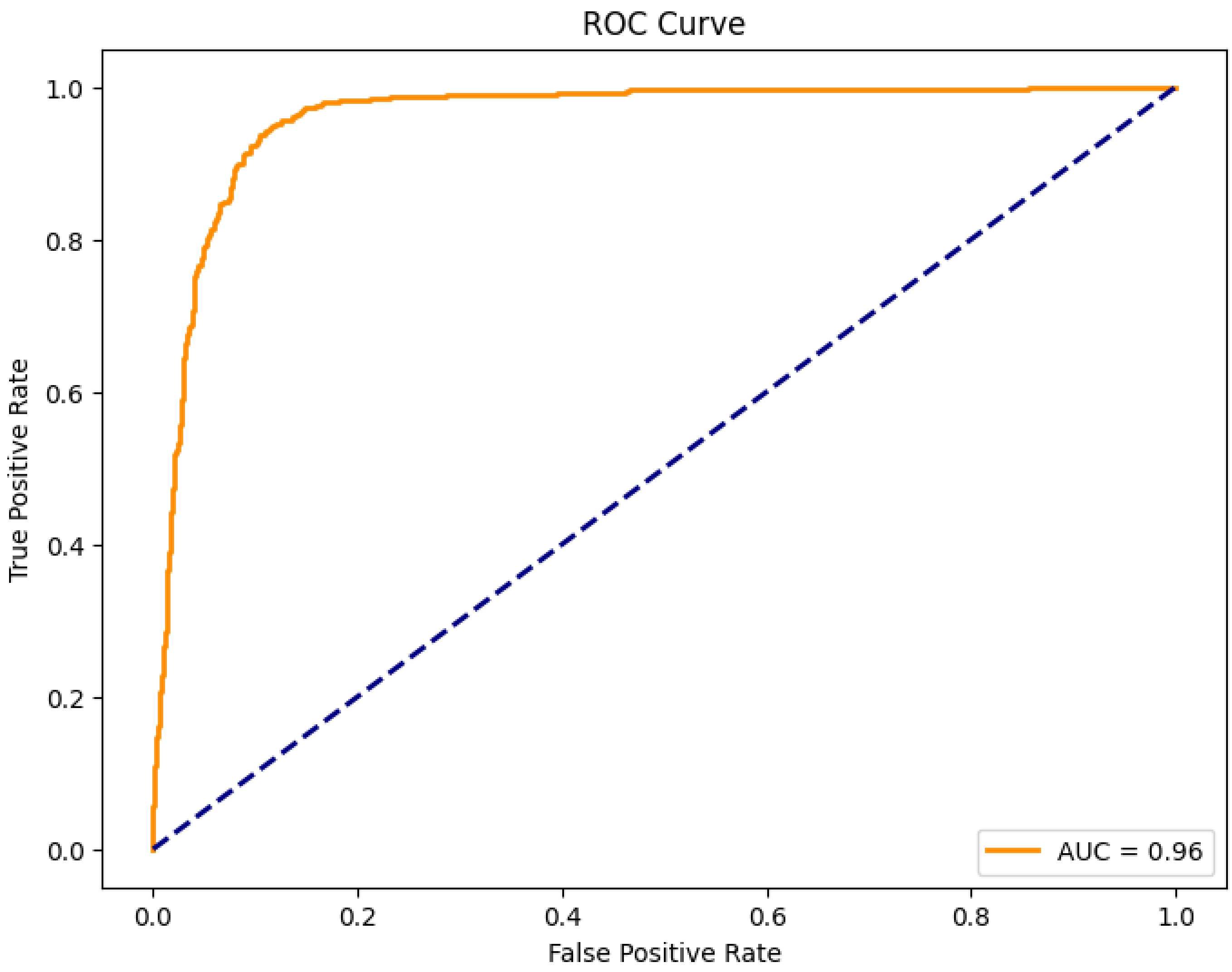
```
# 保存模型的状态字典
model_state_dict = Roberta_Based.state_dict()
```

```
# 保存模型参数
torch.save(model_state_dict, 'roberta_based_model.pth')
```

In [ ]:

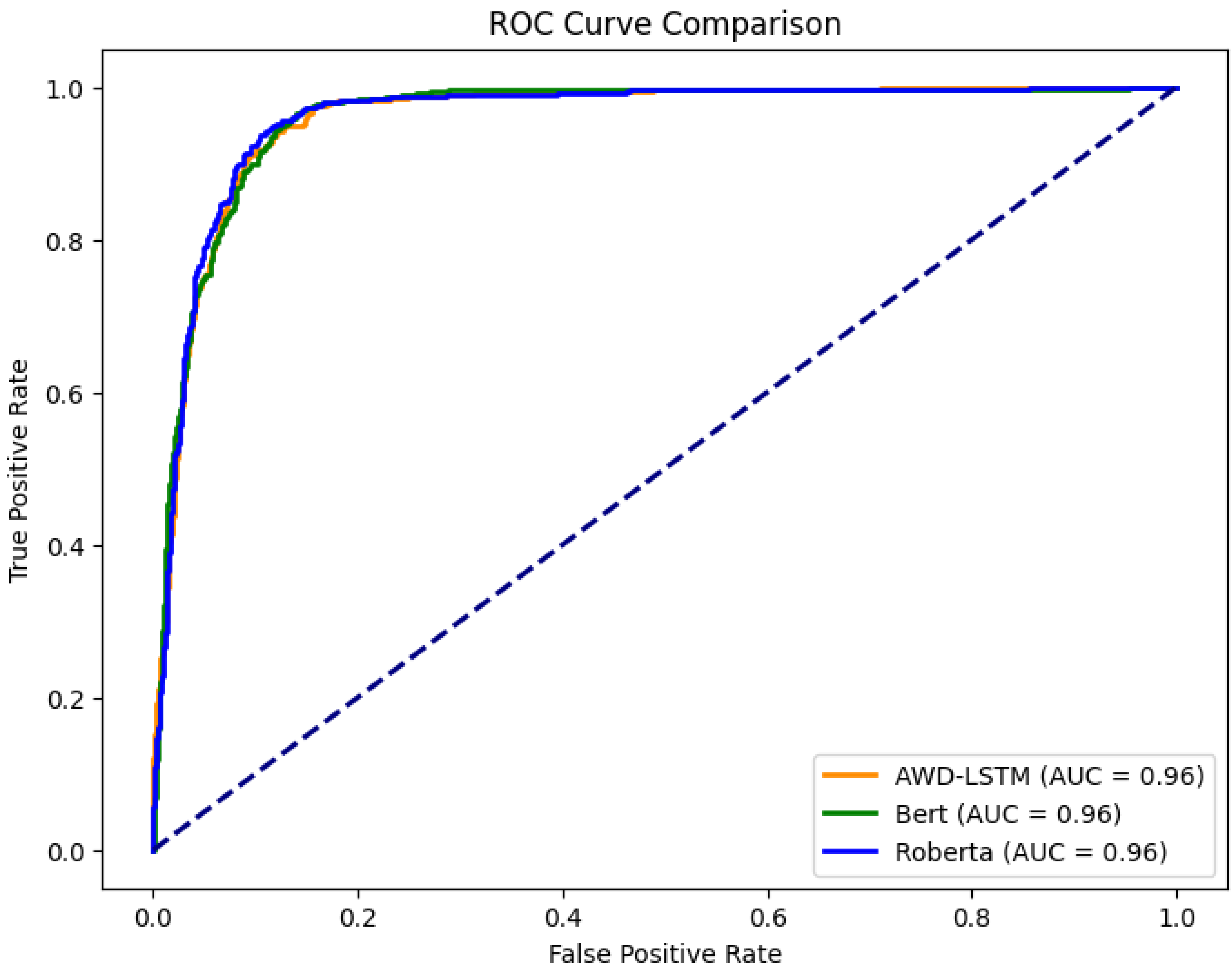
```
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt

# 计算并绘制混淆矩阵
conf_matrix = confusion_matrix(y_val, (ppp3 > 0.5).astype(int))
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], ytic
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# 计算并绘制ROC曲线
fpr, tpr, thresholds = roc_curve(y_val, ppp3)
roc_auc = roc_auc_score(y_val, ppp3)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```





In [ ]:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr1, tpr1, _ = roc_curve(y_val, ppp1)
roc_auc1 = auc(fpr1, tpr1)

fpr2, tpr2, _ = roc_curve(y_val, ppp2)
roc_auc2 = auc(fpr2, tpr2)

fpr3, tpr3, thresholds = roc_curve(y_val, ppp3)
roc_auc3 = auc(fpr3, tpr3)

plt.figure(figsize=(8, 6))
plt.plot(fpr1, tpr1, color='darkorange', lw=2, label=f'AWD-LSTM (AUC = {roc_auc1:.2f})')
plt.plot(fpr2, tpr2, color='green', lw=2, label=f'Bert (AUC = {roc_auc2:.2f})')
plt.plot(fpr3, tpr3, color='blue', lw=2, label=f'Roberta (AUC = {roc_auc3:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend()
plt.show()
```



3 Test

3.1 Data Preparation

In [ ]:

```
X_train = train_df.drop(['label'], axis=1)
y_train = train_df['label']
X_test = test_df
```

In [ ]:

```
text_column = 'text'
if text_column in X_train.columns:
    train_list = X_train[text_column].astype(str).tolist()
else:
    print(f"Error: '{text_column}' column not found in the DataFrame.")

if text_column in X_test.columns:
    test_list = X_test[text_column].astype(str).tolist()
else:
    print(f"Error: '{text_column}' column not found in the DataFrame.")
```

3.2 Model Prediction

In [ ]:

```
# Model 1: AWD-LSTM
from fastai.text.all import *
df_all = pd.concat([train_df.drop(['label'], axis=1),test_df])
TDL_Train = TextDataLoaders.from_df(df_all, text_col=1, is_lm=True)
```

```
TDL_Classify = TextDataLoaders.from_df(
    pd.concat([X_train, y_train], axis=1),
    valid_pct=0.2,
    seed=42,
    text_col=1,
    label_col=2,
    text_vocab=TDL_Train.vocab)

AWD_LSTM_Classifier = text_classifier_learner(TDL_Classify , AWD_LSTM, drop_mult=1, metr
AWD_LSTM_Classifier.path = path
AWD_LSTM_Classifier = AWD_LSTM_Classifier.load_encoder(file='AWD_LSTM_learn')

AWD_LSTM_Classifier.fit_one_cycle(5, slice(1e-4, 1e-2))

AWD_LSTM_Classifier.freeze_to(-2)
AWD_LSTM_Classifier.fit_one_cycle(5, slice(1e-4,1e-2))

AWD_LSTM_Classifier.freeze_to(-3)
AWD_LSTM_Classifier.fit_one_cycle(5, slice(1e-4,1e-2))

AWD_LSTM_Classifier.unfreeze()
AWD_LSTM_Classifier.fit_one_cycle(10, slice(1e-5,1e-3))

## Getting testing data and doing predictions
test_dl = AWD_LSTM_Classifier.dls.test_dl(X_test)
preds, _ = AWD_LSTM_Classifier.get_preds(dl=test_dl)
ppp1 = preds[:, 1]#predicted_positive_probabilities

del TDL_Classify, AWD_LSTM_Classifier, test_dl, preds
```

epoch	train_loss	valid_loss	accuracy	time
0	0.480768	0.335155	0.882576	00:13
1	0.391937	0.252964	0.895833	00:10
2	0.359938	0.245498	0.902146	00:09
3	0.352650	0.243152	0.900253	00:18
4	0.335155	0.236021	0.904672	00:19

epoch	train_loss	valid_loss	accuracy	time
0	0.319513	0.234250	0.903409	00:13
1	0.335817	0.251796	0.897727	00:14
2	0.320423	0.249445	0.898990	00:18
3	0.303719	0.233720	0.901515	00:19
4	0.281017	0.238421	0.900253	00:16

epoch	train_loss	valid_loss	accuracy	time
0	0.267715	0.250202	0.909091	00:11
1	0.293601	0.265992	0.885101	00:11
2	0.278400	0.264548	0.897727	00:11
3	0.255770	0.241518	0.904672	00:09

4	0.257977	0.235698	0.902778	00:11

epoch	train_loss	valid_loss	accuracy	time
0	0.237903	0.234497	0.904672	00:12
1	0.242053	0.229372	0.907828	00:12
2	0.235835	0.229236	0.910354	00:10
3	0.226488	0.238888	0.905303	00:12
4	0.228484	0.240075	0.904672	00:12
5	0.225377	0.248998	0.902778	00:10
6	0.229425	0.237226	0.905934	00:11
7	0.221833	0.237807	0.907828	00:12
8	0.226278	0.243778	0.904672	00:10
9	0.222860	0.241383	0.907828	00:11

In [ ]:

```
df = pd.DataFrame(ppp1, columns=['ppp1'])
df.to_csv('ppp1.csv', index=False)
```

In [ ]:

```
# Model 2: Bert
from transformers import BertTokenizer, BertForSequenceClassification

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
Bert_Based = BertForSequenceClassification.from_pretrained('bert-base-uncased')

inputs = tokenizer(train_list, return_tensors="pt", truncation=True, padding=True)
labels = torch.tensor(y_train.tolist(), dtype=torch.float32)

import torch
from torch.utils.data import DataLoader, TensorDataset
from tqdm import tqdm
Bert_Based.to('cuda')
inputs_cuda = {key: value.to('cuda') for key, value in inputs.items()}
labels_tensor_cuda = labels.to('cuda')
dataset = TensorDataset(inputs_cuda['input_ids'], inputs_cuda['attention_mask'], labels_
dataloader = DataLoader(dataset, batch_size=128, shuffle=True)
optimizer = torch.optim.AdamW(Bert_Based.parameters(), lr=1e-5)
criterion = torch.nn.CrossEntropyLoss()
train_loss_history = []
train_accuracy_history = []

for epoch in range(5):
    total_loss = 0.0
    total_correct_predictions = 0
    total_samples = 0

    progress_bar = tqdm(enumerate(dataloader), total=len(dataloader), desc=f'Epoch {epoc

    for batch_idx, batch in progress_bar:
        optimizer.zero_grad()
        outputs = Bert_Based(input_ids=batch[0], attention_mask=batch[1])
        logits = outputs.logits
        loss = criterion(logits, batch[2].long())

        loss.backward()
        optimizer.step()
        total_loss += loss.item()

        _, predicted_labels = torch.max(logits, 1)
        total_correct_predictions += (predicted_labels == batch[2]).sum().item()
        total_samples += batch[2].size(0)

        progress_bar.set_postfix({'Loss': loss.item(), 'Accuracy': total_correct_predict

    epoch_loss = total_loss / len(dataloader)
    epoch_accuracy = total_correct_predictions / total_samples

    train_loss_history.append(epoch_loss)
    train_accuracy_history.append(epoch_accuracy)

    print(f'Epoch {epoch + 1}/{3}, Average Loss: {epoch_loss:.4f}, Average Accuracy: {ep

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
test_tokenized = tokenizer(test_list, padding=True, truncation=True, return_tensors='pt'
test_list_cuda = {key: value.to('cuda') for key, value in test_tokenized.items()}

Bert_Based.eval()
with torch.no_grad():
    outputs_test = Bert_Based(input_ids=test_list_cuda['input_ids'], attention_mask=test_
    logits_test = outputs_test.logits
ppp2 = torch.softmax(logits_test, dim=1)[:, 1].cpu().numpy()

del tokenizer, Bert_Based, inputs, labels
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:

The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://
huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your
session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public model
s or datasets.
  warnings.warn(
Epoch 1/5: 100%|██████████| 62/62 [00:23<00:00,  2.60it/s, Loss=0.251, Accuracy=0.865]
Epoch 1/3, Average Loss: 0.3154, Average Accuracy: 0.8654
Epoch 2/5: 100%|██████████| 62/62 [00:23<00:00,  2.60it/s, Loss=0.247, Accuracy=0.927]
Epoch 2/3, Average Loss: 0.1903, Average Accuracy: 0.9274
Epoch 3/5: 100%|██████████| 62/62 [00:23<00:00,  2.60it/s, Loss=0.173, Accuracy=0.958]
Epoch 3/3, Average Loss: 0.1324, Average Accuracy: 0.9576
Epoch 4/5: 100%|██████████| 62/62 [00:23<00:00,  2.60it/s, Loss=0.0362, Accuracy=0.977]
Epoch 4/3, Average Loss: 0.0858, Average Accuracy: 0.9765
Epoch 5/5: 100%|██████████| 62/62 [00:23<00:00,  2.60it/s, Loss=0.0627, Accuracy=0.987]
Epoch 5/3, Average Loss: 0.0548, Average Accuracy: 0.9866
```

In [ ]:

```
df = pd.DataFrame(ppp2, columns=['ppp2'])
df.to_csv('ppp2.csv', index=False)
```

In [ ]:

```
# Model 3: Roberta
from transformers import RobertaTokenizer, RobertaForSequenceClassification
from torch.utils.data import DataLoader, TensorDataset
import torch

tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
Roberta_Based = RobertaForSequenceClassification.from_pretrained('roberta-base', num_lab

inputs = tokenizer(train_list, return_tensors="pt", truncation=True, padding=True)
labels = torch.tensor(y_train.tolist(), dtype=torch.float32)
```

```
from torch.utils.data import DataLoader, TensorDataset
from tqdm import tqdm

Roberta_Based.to('cuda')
inputs_cuda = {key: value.to('cuda') for key, value in inputs.items()}
labels_tensor_cuda = labels.to('cuda')

dataset = TensorDataset(inputs_cuda['input_ids'], inputs_cuda['attention_mask'], labels_
dataloader = DataLoader(dataset, batch_size=256, shuffle=True)

optimizer = torch.optim.AdamW(Roberta_Based.parameters(), lr=1e-5)
criterion = torch.nn.CrossEntropyLoss()

train_loss_history = []
train_accuracy_history = []

for epoch in range(5):
    total_loss = 0.0
    total_correct_predictions = 0
    total_samples = 0

    progress_bar = tqdm(enumerate(dataloader), total=len(dataloader), desc=f'Epoch {epoc

    for batch_idx, batch in progress_bar:
        optimizer.zero_grad()
        outputs = Roberta_Based(input_ids=batch[0], attention_mask=batch[1])
        logits = outputs.logits
        loss = criterion(logits, batch[2].long())

        loss.backward()
        optimizer.step()

        total_loss += loss.item()

        _, predicted_labels = torch.max(logits, 1)
        total_correct_predictions += (predicted_labels == batch[2]).sum().item()
        total_samples += batch[2].size(0)

        progress_bar.set_postfix({'Loss': loss.item(), 'Accuracy': total_correct_predict

    epoch_loss = total_loss / len(dataloader)
    epoch_accuracy = total_correct_predictions / total_samples

    train_loss_history.append(epoch_loss)
    train_accuracy_history.append(epoch_accuracy)

    print(f'Epoch {epoch + 1}/5, Average Loss: {epoch_loss:.4f}, Average Accuracy: {epoc

tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
test_tokenized = tokenizer(test_list, padding=True, truncation=True, return_tensors='pt'
test_list_cuda = {key: value.to('cuda') for key, value in test_tokenized.items()}

# 开始验证
Roberta_Based.eval()
with torch.no_grad():
    outputs_test = Roberta_Based(input_ids=test_list_cuda['input_ids'], attention_mask=t
    logits_test = outputs_test.logits
ppp3 = torch.softmax(logits_test, dim=1)[:, 1].cpu().numpy()
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:

The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://
huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your
session.
```

```
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public model
s or datasets.
  warnings.warn(
Epoch 1/5: 100%|██████████| 31/31 [00:29<00:00,  1.04it/s, Loss=0.319, Accuracy=0.728]
Epoch 1/5, Average Loss: 0.4659, Average Accuracy: 0.7277
Epoch 2/5: 100%|██████████| 31/31 [00:26<00:00,  1.19it/s, Loss=0.243, Accuracy=0.877]
Epoch 2/5, Average Loss: 0.2550, Average Accuracy: 0.8774
Epoch 3/5: 100%|██████████| 31/31 [00:25<00:00,  1.20it/s, Loss=0.164, Accuracy=0.923]
Epoch 3/5, Average Loss: 0.1887, Average Accuracy: 0.9235
Epoch 4/5: 100%|██████████| 31/31 [00:25<00:00,  1.20it/s, Loss=0.162, Accuracy=0.94]
Epoch 4/5, Average Loss: 0.1548, Average Accuracy: 0.9402
Epoch 5/5: 100%|██████████| 31/31 [00:25<00:00,  1.20it/s, Loss=0.069, Accuracy=0.954]
Epoch 5/5, Average Loss: 0.1251, Average Accuracy: 0.9535
```

In [ ]:

```
df = pd.DataFrame(ppp3, columns=['ppp3'])
df.to_csv('ppp3.csv', index=False)
```

3.3 Ensemble Results

In [ ]:

```
ppp1 = pd.read_csv(path/'ppp1.csv')
ppp2 = pd.read_csv(path/'ppp2.csv')
ppp3 = pd.read_csv(path/'ppp3.csv')

result1 = (ppp1 > 0.5).astype(int)
result2 = (ppp2 > 0.5).astype(int)
result3 = (ppp3 > 0.5).astype(int)
```

In [ ]:

```
#Ordinary Averaging
average_array = (ppp1 + ppp2 + ppp3) / 3
Average_result = (average_array > 0.5).astype(int)
Average_result
```

Out[ ]:

	ppp1
0	1
1	1
2	1
3	1
4	1
...	...
1948	0
1949	0
1950	1
1951	0
1952	0

1953 rows × 1 columns

In [ ]:

```
#Hard Voting
vote_result = np.sum([result1, result2, result3], axis=0) > 1
vote_result = vote_result.astype(int)
vote_result
```

Out[ ]:

```
array([[1],
       [1],
       [1],
       ...,
       [1],
       [0],
       [0]])
```

4 Results

In [ ]:

```
submission_df = pd.read_csv(path/'sample_submission_LnhVWA4.csv')
submission_df
```

Out[ ]:

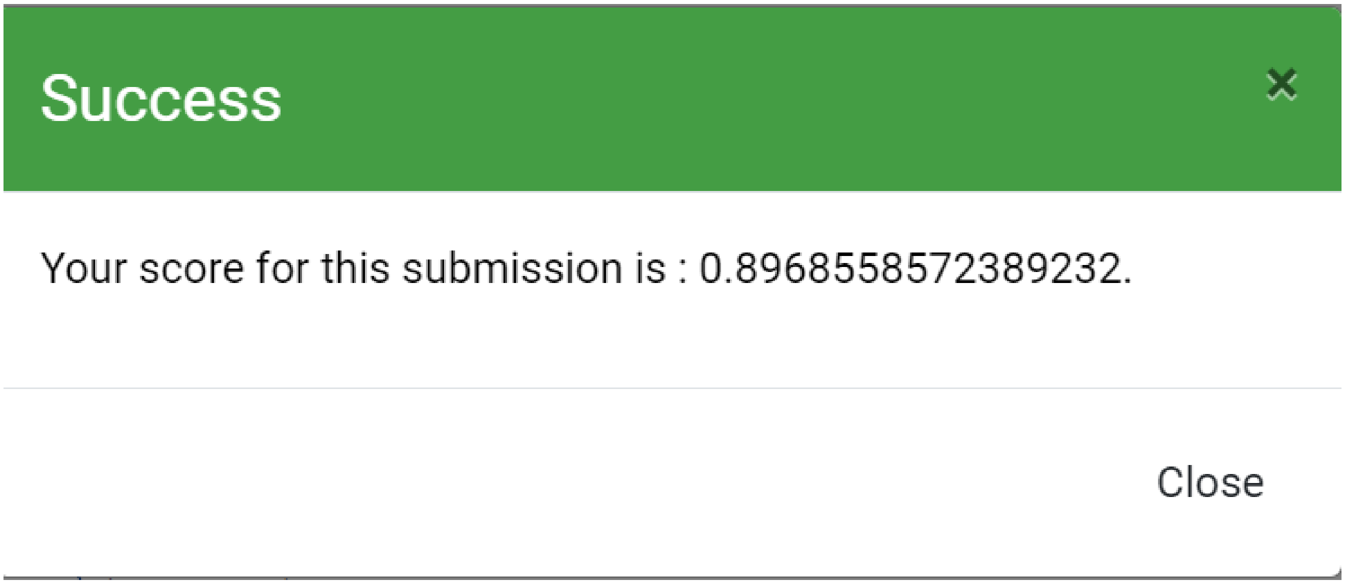
	id	label
0	7921	0
1	7922	0
2	7923	0
3	7924	0
4	7925	0
...	...	...
1948	9869	0
1949	9870	0
1950	9871	0
1951	9872	0
1952	9873	0

1953 rows × 2 columns

4.1 Individual Model

In [ ]:

```
AWD_LSTM_Result = submission_df
AWD_LSTM_Result['label'] = result1
AWD_LSTM_Result.to_csv('AWD_LSTM_Result.csv', index=False)
```

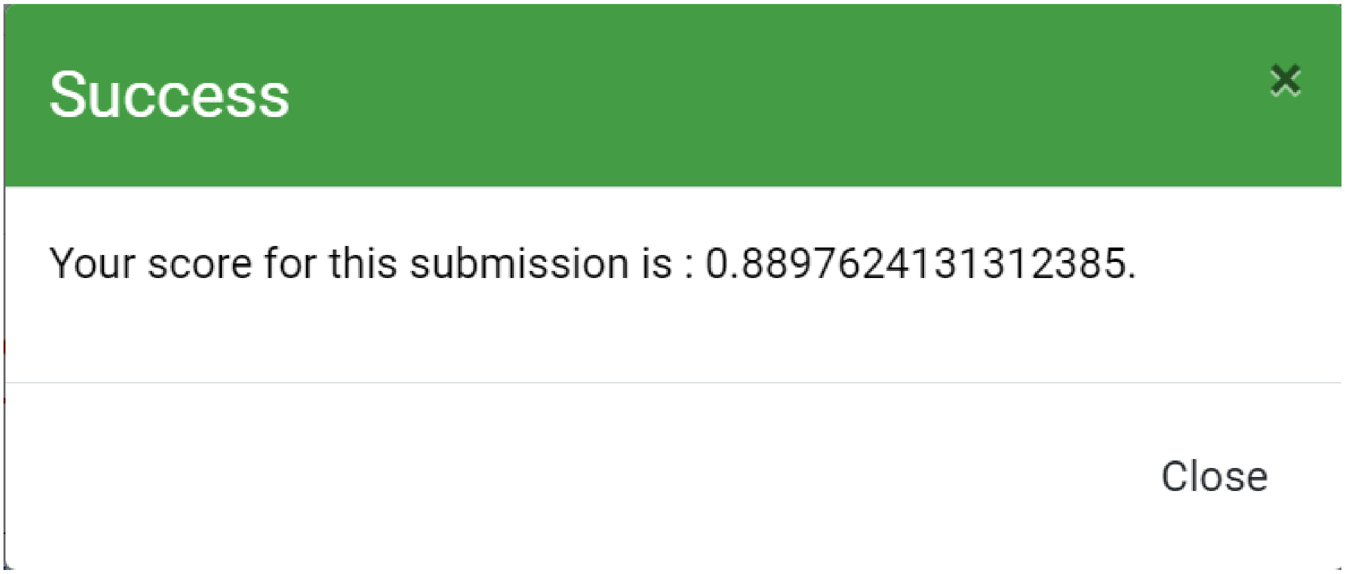


Success

Your score for this submission is : 0.8968558572389232.

In [ ]:

```
Bert_Result = submission_df
Bert_Result['label'] = result2
Bert_Result.to_csv('Bert_Result.csv', index=False)
```

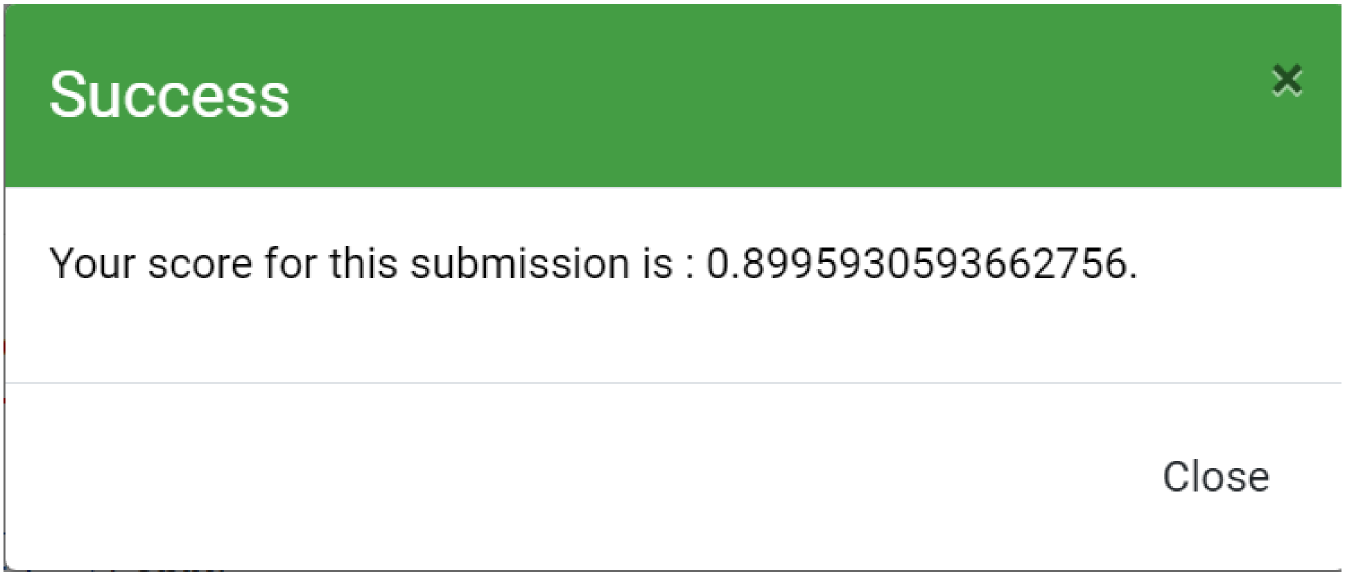


Success

Your score for this submission is : 0.8897624131312385.

In [ ]:

```
Roberta_Result = submission_df
Roberta_Result['label'] = result3
Roberta_Result.to_csv('Roberta_Result.csv', index=False)
```



Success

Your score for this submission is : 0.8995930593662756.

4.2 Ensemble Results:

In [ ]:

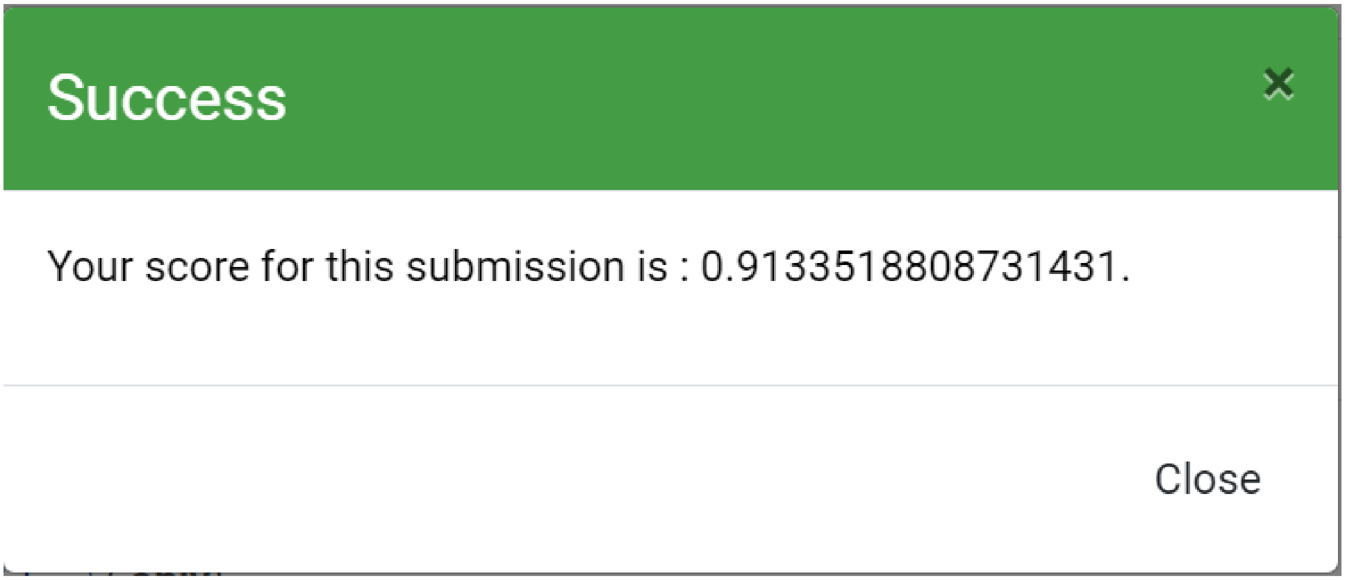
```
#Ordinary Averaging
Average_Result = submission_df
Average_Result['label'] = Average_result['ppp1']
Average_Result.to_csv('Average_result.csv', index=False)
Average_Result
```

Out[ ]:

	id	label
0	7921	1
1	7922	1
2	7923	1
3	7924	1

4	7925	1
...	...	...
1948	9869	0
1949	9870	0
1950	9871	1
1951	9872	0
1952	9873	0

1953 rows × 2 columns



Success

Your score for this submission is : 0.9133518808731431.

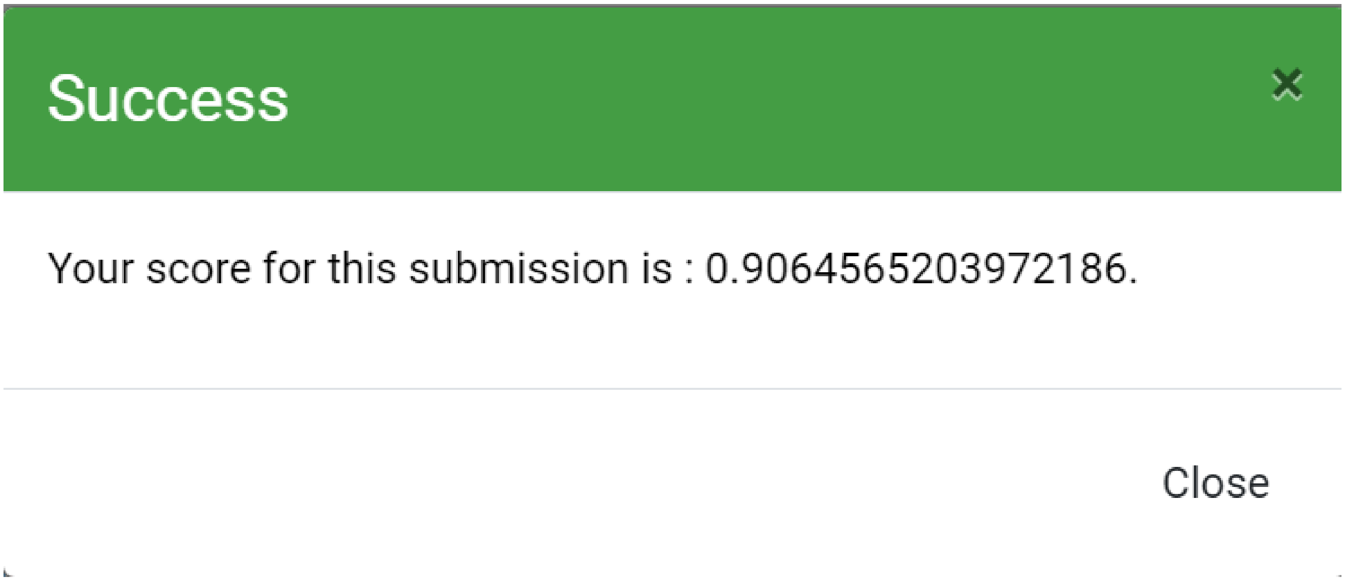
In [ ]:

```
#Hard Voting
Vote_Result = submission_df
Vote_Result['label'] = vote_result
Vote_Result.to_csv('Vote_result.csv', index=False)
Vote_Result
```

Out[ ]:

	id	label
0	7921	1
1	7922	1
2	7923	1
3	7924	1
4	7925	1
...	...	...
1948	9869	0
1949	9870	0
1950	9871	1
1951	9872	0
1952	9873	0

1953 rows × 2 columns

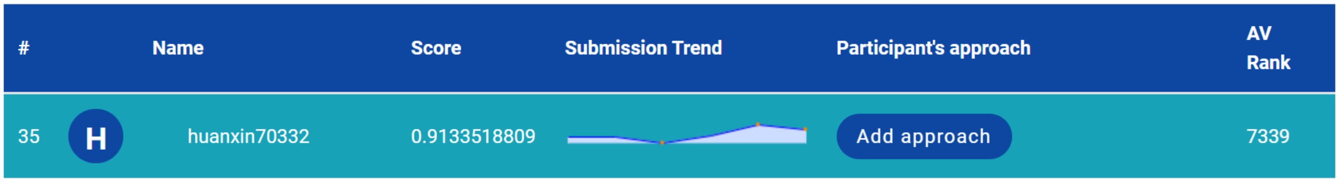


Success

Your score for this submission is : 0.9064565203972186.

4.3 Final Rank

https://datahack.analyticsvidhya.com/contest/linguipedia-codefest-natural-language-processing-1/#LeaderBoard



#		Name	Score	Submission Trend	Participant's approach	AV Rank
35	H	huanxin70332	0.9133518809		Add approach	7339