3.6.1-bc-snc-single Lexical Complexity Binary Classification Prediction Transformers Modeling

April 13, 2025

0.1 Packages, Library Imports, File Mounts, & Data Imports ** Run All **

[69]:
```
# @title
!pip install -q transformers
!pip install -q torchinfo
!pip install -q datasets
!pip install -q evaluate
!pip install -q nltk
!pip install -q contractions
!pip install -q hf_xet
!pip install -q sentencepiece
```

[70]:
```
!sudo apt-get update
! sudo apt-get install tree
```

```
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:6 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64
InRelease
Hit:7 https://r2u.stat.illinois.edu/ubuntu jammy InRelease
Hit:8 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:9 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy
InRelease
Hit:10 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Reading package lists… Done
W: Skipping acquire of configured file 'main/source/Sources' as repository
'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem to provide
it (sources.list entry misspelt?)
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
tree is already the newest version (2.0.2-1).
0 upgraded, 0 newly installed, 0 to remove and 32 not upgraded.
```

[71]:
```
#@title Imports
import nltk
from nltk.tokenize import RegexpTokenizer
import sentencepiece
import contractions
import spacy


import evaluate
from datasets import load_dataset, Dataset, DatasetDict

import torch
import torch.nn as nn
from torchinfo import summary

import transformers
from transformers import AutoTokenizer, AutoModel,␣
 ↪AutoModelForSequenceClassification, TrainingArguments, Trainer, BertConfig,␣
 ↪BertForSequenceClassification

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import sklearn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report,␣
 ↪precision_recall_fscore_support, accuracy_score

import json
import datetime
import zoneinfo
from datetime import datetime
```

[72]:
```
# @title Mount Google Drive
```

[73]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

[74]:
```
dir_root = '/content/drive/MyDrive/266-final/'
# dir_data = '/content/drive/MyDrive/266-final/data/'
```

```
# dir_data = '/content/drive/MyDrive/266-final/data/se21-t1-comp-lex-master/'
dir_data = '/content/drive/MyDrive/266-final/data/266-comp-lex-master'
dir_models = '/content/drive/MyDrive/266-final/models/'
dir_results = '/content/drive/MyDrive/266-final/results/'
log_filename = "experiment_runs.txt"
log_filepath = os.path.join(dir_results, log_filename)
```

[75]:
```
wandbai_api_key = ""
```

[76]:
```
!tree /content/drive/MyDrive/266-final/data/266-comp-lex-master/
```

```
/content/drive/MyDrive/266-final/data/266-comp-lex-master/
  fe-test-labels
      test_multi_df.csv
      test_single_df.csv
  fe-train
      train_multi_df.csv
      train_single_df.csv
  fe-trial-val
      trial_val_multi_df.csv
      trial_val_single_df.csv
  test-labels
      lcp_multi_test.tsv
      lcp_single_test.tsv
  train
      lcp_multi_train.tsv
      lcp_single_train.tsv
  trial
       lcp_multi_trial.tsv
       lcp_single_trial.tsv

6 directories, 12 files
```

[77]:
```
!ls -R /content/drive/MyDrive/266-final/data/266-comp-lex-master/
```

```
/content/drive/MyDrive/266-final/data/266-comp-lex-master/:
fe-test-labels  fe-train  fe-trial-val  test-labels  train  trial

/content/drive/MyDrive/266-final/data/266-comp-lex-master/fe-test-labels:
test_multi_df.csv  test_single_df.csv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/fe-train:
train_multi_df.csv  train_single_df.csv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/fe-trial-val:
trial_val_multi_df.csv  trial_val_single_df.csv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/test-labels:
```

```
lcp_multi_test.tsv  lcp_single_test.tsv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/train:
lcp_multi_train.tsv  lcp_single_train.tsv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/trial:
lcp_multi_trial.tsv  lcp_single_trial.tsv
```

[78]:
```
!tree /content/drive/MyDrive/266-final/data/266-comp-lex-master/
```

```
/content/drive/MyDrive/266-final/data/266-comp-lex-master/
  fe-test-labels
      test_multi_df.csv
      test_single_df.csv
  fe-train
      train_multi_df.csv
      train_single_df.csv
  fe-trial-val
      trial_val_multi_df.csv
      trial_val_single_df.csv
  test-labels
      lcp_multi_test.tsv
      lcp_single_test.tsv
  train
      lcp_multi_train.tsv
      lcp_single_train.tsv
  trial
       lcp_multi_trial.tsv
       lcp_single_trial.tsv

6 directories, 12 files
```

[79]:
```
#@title Import Data
```

[80]:
```
df_names = [
    "train_single_df",
    "train_multi_df",
    "trial_val_single_df",
    "trial_val_multi_df",
    "test_single_df",
    "test_multi_df"
]

loaded_dataframes = {}

for df_name in df_names:
    if "train" in df_name:
        subdir = "fe-train"
```

```
    elif "trial_val" in df_name:
        subdir = "fe-trial-val"
    elif "test" in df_name:
        subdir = "fe-test-labels"
    else:
        subdir = None

    if subdir:
        read_path = os.path.join(dir_data, subdir, f"{df_name}.csv")
        loaded_df = pd.read_csv(read_path)
        loaded_dataframes[df_name] = loaded_df
        print(f"Loaded {df_name} from {read_path}")

# for df_name, df in loaded_dataframes.items():
#     print(f"\n>>> {df_name} shape: {df.shape}")
#     if 'binary_complexity' in df.columns:
#         print(df['binary_complexity'].value_counts())
#         print(df.info())
#         print(df.head())

for df_name, df in loaded_dataframes.items():
    globals()[df_name] = df
    print(f"{df_name} loaded into global namespace.")
```

```
Loaded train_single_df from /content/drive/MyDrive/266-final/data/266-comp-lex-
master/fe-train/train_single_df.csv
Loaded train_multi_df from /content/drive/MyDrive/266-final/data/266-comp-lex-
master/fe-train/train_multi_df.csv
Loaded trial_val_single_df from /content/drive/MyDrive/266-final/data/266-comp-
lex-master/fe-trial-val/trial_val_single_df.csv
Loaded trial_val_multi_df from /content/drive/MyDrive/266-final/data/266-comp-
lex-master/fe-trial-val/trial_val_multi_df.csv
Loaded test_single_df from /content/drive/MyDrive/266-final/data/266-comp-lex-
master/fe-test-labels/test_single_df.csv
Loaded test_multi_df from /content/drive/MyDrive/266-final/data/266-comp-lex-
master/fe-test-labels/test_multi_df.csv
train_single_df loaded into global namespace.
train_multi_df loaded into global namespace.
trial_val_single_df loaded into global namespace.
trial_val_multi_df loaded into global namespace.
test_single_df loaded into global namespace.
test_multi_df loaded into global namespace.
```

- Functional tests pass, we can proceed with Baseline Modeling

0.2 Experiments

0.2.1 Helper Functions ** Run **

[81]:
```
MODEL_LINEAGE = {}

def get_model_and_tokenizer(
    remote_model_name: str = None,
    local_model_path: str = None,
    config=None
):
    """
    Loads the model & tokenizer for classification.
    If 'local_model_path' is specified, load from that path.
    Otherwise, fall back to 'remote_model_name'.

    Optional: 'config' can be a custom BertConfig/AutoConfig object
              to override certain configuration parameters.

    Records complete traceable lineage in the global MODEL_LINEAGE.
    """
    global MODEL_LINEAGE

    if local_model_path:
        print(f"Loading from local path: {local_model_path}")
        tokenizer = AutoTokenizer.from_pretrained(local_model_path)

        # If a config object is provided, we pass it to from_pretrained.
        # Otherwise, it just uses the config that is part of local_model_path.
        if config is not None:
            model = AutoModelForSequenceClassification.from_pretrained(
                local_model_path,
                config=config
            )
        else:
            model = AutoModelForSequenceClassification.
 ↪from_pretrained(local_model_path)

        MODEL_LINEAGE = {
            "type": "offline_checkpoint",
            "path": local_model_path,
            "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        }
    elif remote_model_name:
        print(f"Loading from Hugging Face model: {remote_model_name}")
        tokenizer = AutoTokenizer.from_pretrained(remote_model_name)

        if config is not None:
```

```
            model = AutoModelForSequenceClassification.from_pretrained(
                remote_model_name,
                config=config
            )
        else:
            model = AutoModelForSequenceClassification.
 ↪from_pretrained(remote_model_name)

        MODEL_LINEAGE = {
            "type": "huggingface_hub",
            "path": remote_model_name,
            "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        }
    else:
        raise ValueError("You must provide either a remote_model_name or a␣
 ↪local_model_path!")

    return model, tokenizer
```

[82]:
```
def freeze_unfreeze_layers(model, layers_to_unfreeze=None):
    """
    Toggles requires_grad = False for all parameters
    except for those whose names contain any string in layers_to_unfreeze.
    By default, always unfreeze classifier/heads.
    """
    if layers_to_unfreeze is None:
        layers_to_unfreeze = ["classifier.", "pooler."]

    for name, param in model.named_parameters():
        if any(substring in name for substring in layers_to_unfreeze):
            param.requires_grad = True
        else:
            param.requires_grad = False
```

[83]:
```
def encode_examples(examples, tokenizer, text_col, max_length=256):
    """
    Tokenizes a batch of texts from 'examples[text_col]' using the given␣
 ↪tokenizer.
    Returns a dict with 'input_ids', 'attention_mask', etc.
    """
    texts = examples[text_col]
    encoded = tokenizer(
        texts,
        truncation=True,
        padding='max_length',
        max_length=max_length
    )
```

```
    return encoded
```

[84]:
```
def prepare_dataset(df, tokenizer, text_col, label_col, max_length=256):
    """
    Converts a Pandas DataFrame to a Hugging Face Dataset,
    then applies 'encode_examples' to tokenize.
    """
    dataset = Dataset.from_pandas(df)

    dataset = dataset.map(
        lambda batch: encode_examples(batch, tokenizer, text_col, max_length),
        batched=True
    )

    dataset = dataset.rename_column(label_col, "labels")
    dataset.set_format(type='torch',
                       columns=['input_ids', 'attention_mask', 'labels'])
    return dataset
```

[85]:
```
def compute_metrics(eval_pred):
    """
    Computes classification metrics, including accuracy, precision, recall, and␣
 ↪F1.
    """
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=1)

    metric_accuracy  = evaluate.load("accuracy")
    metric_precision = evaluate.load("precision")
    metric_recall    = evaluate.load("recall")
    metric_f1        = evaluate.load("f1")

    accuracy_result  = metric_accuracy.compute(predictions=preds,␣
 ↪references=labels)
    precision_result = metric_precision.compute(predictions=preds,␣
 ↪references=labels, average="binary")
    recall_result    = metric_recall.compute(predictions=preds,␣
 ↪references=labels, average="binary")
    f1_result        = metric_f1.compute(predictions=preds, references=labels,␣
 ↪average="binary")

    return {
        "accuracy"       : accuracy_result["accuracy"],
        "precision": precision_result["precision"],
        "recall"   : recall_result["recall"],
        "f1"       : f1_result["f1"]
    }
```

[86]:
```
def gather_config_details(model):
    """
    Enumerates every attribute in model.config
    """
    config_items = {}
    for attr_name, attr_value in vars(model.config).items():
        config_items[attr_name] = attr_value
    return config_items

def gather_model_details(model):
    """
    Extracts total layers, total params, trainable params, and activation␣
 ↪function
    from a Transformers model. Adjust logic as needed for different␣
 ↪architectures.
    """
    details = {}

    try:
        total_params = model.num_parameters()
        trainable_params = model.num_parameters(only_trainable=True)
    except AttributeError:
        all_params = list(model.parameters())
        total_params = sum(p.numel() for p in all_params)
        trainable_params = sum(p.numel() for p in all_params if p.requires_grad)

    details["model_total_params"] = total_params
    details["model_trainable_params"] = trainable_params

    if hasattr(model, "bert") and hasattr(model.bert, "pooler"):
        act_obj = getattr(model.bert.pooler, "activation", None)
        details["pooler_activation_function"] = act_obj.__class__.__name__ if␣
 ↪act_obj else "N/A"
    else:
        details["pooler_activation_function"] = "N/A"

    details["config_attributes"] = gather_config_details(model)
    return details

def gather_all_run_metrics(trainer, train_dataset=None, val_dataset=None,␣
 ↪test_dataset=None):
    """
    Gathers final training metrics, final validation metrics, final test␣
 ↪metrics.
    Instead of only parsing the final train_loss from the log, we also do a full
    trainer.evaluate(train_dataset) to get the same set of metrics that val/
 ↪test have.
```

```
    """
    results = {}

    if train_dataset is not None:
        train_metrics = trainer.evaluate(train_dataset)
        for k, v in train_metrics.items():
            results[f"train_{k}"] = v
    else:
        results["train_metrics"] = "No train dataset provided"

    if val_dataset is not None:
        val_metrics = trainer.evaluate(val_dataset)
        for k, v in val_metrics.items():
            results[f"val_{k}"] = v
    else:
        results["val_metrics"] = "No val dataset provided"

    if test_dataset is not None:
        test_metrics = trainer.evaluate(test_dataset)
        for k, v in test_metrics.items():
            results[f"test_{k}"] = v
    else:
        results["test_metrics"] = "No test dataset provided"

    return results

# def log_experiment_results_json(experiment_meta, model_details, run_metrics,␣
 ↪log_file):
#     """
#     Logs experiment metadata, model details, and metrics to a JSON lines file.
#     Automatically concatenates the 'checkpoint_path' to the 'model_lineage'.
#     """
#     checkpoint_path = model_details.get("checkpoint_path")
#     if checkpoint_path:
#         if "model_lineage" not in model_details:
#             model_details["model_lineage"] = ""
#         if model_details["model_lineage"]:
#             model_details["model_lineage"] += " -> "
#         model_details["model_lineage"] += checkpoint_path

#     record = {
#         "timestamp": str(datetime.datetime.now()),
#         "experiment_meta": experiment_meta,
#         "model_details": model_details,
#         "run_metrics": run_metrics
#     }
```

```
#     with open(log_file, "a", encoding="utf-8") as f:
#         json.dump(record, f)
#         f.write("\n")

def log_experiment_results_json(experiment_meta, model_details, run_metrics,␣
 ↪log_file):
    """
    Logs experiment metadata, model details, and metrics to a JSON lines file.
    Automatically concatenates the 'checkpoint_path' to the 'model_lineage'
    and uses Pacific time for the timestamp.
    """
    checkpoint_path = model_details.get("checkpoint_path")
    if checkpoint_path:
        if "model_lineage" not in model_details:
            model_details["model_lineage"] = ""
        if model_details["model_lineage"]:
            model_details["model_lineage"] += " -> "
        model_details["model_lineage"] += checkpoint_path

    pacific_time = datetime.now(zoneinfo.ZoneInfo("America/Los_Angeles")) #␣
 ↪update to support pacific time
    timestamp_str = pacific_time.isoformat()

    record = {
        "timestamp": timestamp_str,
        "experiment_meta": experiment_meta,
        "model_details": model_details,
        "run_metrics": run_metrics
    }

    with open(log_file, "a", encoding="utf-8") as f:
        json.dump(record, f)
        f.write("\n")
```

0.2.2 Experiment Cohort Design

[87]:
```
# Define Experiment Parameters

named_model = "bert-base-cased"
# named_model = "roberta-base"
# named_model = "bert-large"
# named_model = "roberta-large"
# named_model = "" # modern bert

# learning_rate = 1e-3
# learning_rate = 1e-4
learning_rate = 1e-5
```

```
# learning_rate = 5e-6
# learning_rate = 5e-7
# learning_rate = 5e-8

# num_epochs = 1
# num_epochs = 3
# num_epochs = 5
num_epochs = 25
# num_epochs = 15
# num_epochs = 20

# length_max = 128
length_max = 256
# length_max = 348
# length_max = 512

# size_batch = 1
# size_batch = 4
# size_batch = 8
size_batch = 16
# size_batch = 24
# size_batch = 32
# size_batch = 64
# size_batch = 128

# regularization_weight_decay = 0
regularization_weight_decay = 0.1
# regularization_weight_decay = 0.5

y_col = "binary_complexity"
# y_col = "complexity"

x_task = "single"
# x_task = "multi"

# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"

if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
```

```
    df_val   = trial_val_multi_df
    df_test  = test_multi_df

custom_config = BertConfig.from_pretrained("bert-base-cased")

custom_config.hidden_dropout_prob = 0.1
# custom_config.intermediate_size = 3072
# custom_config.intermediate_size = 6144
# custom_config.num_attention_heads = 12
# custom_config.num_hidden_layers = 12
custom_config.gradient_checkpointing = False
custom_config.attention_probs_dropout_prob = 0.1
# custom_config.max_position_embeddings = 512
# custom_config.type_vocab_size = 2
custom_config.hidden_act = "gelu"  # alts: "relu" "silu"
# custom_config.vocab_size = 28996  # must match

# model.bert.pooler.activation = nn.ReLU() # Tanh() replaced as the pooler␣
 ↪layer activation function in side-by-side with 1.1
```

[88]:
```
def train_transformer_model(
    model,
    tokenizer,
    train_dataset,
    val_dataset,
    output_dir=dir_results,
    num_epochs=num_epochs,
    batch_size=size_batch,
    lr=learning_rate,
    weight_decay=regularization_weight_decay
):
    """
    Sets up a Trainer and trains the model for 'num_epochs' using the given␣
 ↪dataset.
    Returns the trained model and the Trainer object for possible re-use or␣
 ↪analysis.
    """

    training_args = TrainingArguments(
        output_dir=output_dir,
        num_train_epochs=num_epochs,
        per_device_train_batch_size=batch_size,
        per_device_eval_batch_size=batch_size,
        evaluation_strategy="epoch",
        save_strategy="no",
        logging_strategy="epoch",
        learning_rate=lr,
```

```
        weight_decay=weight_decay,
        report_to=["none"],  # or "wandb"
        warmup_steps=8
    )

    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=val_dataset,
        tokenizer=tokenizer,  # optional
        compute_metrics=compute_metrics
    )

    trainer.train()
    return model, trainer
```

Model Inspection ** Run **

[89]:
```
print("model checkpoints:", dir_models)
!ls /content/drive/MyDrive/266-final/models/
```

```
model checkpoints: /content/drive/MyDrive/266-final/models/
multi_answerdotai
multi_bert-base-cased_binary_complexity_20250408_143322
multi_bert-base-cased_binary_complexity_20250409_175804
multi_bert-base-cased_binary_complexity_20250409_175954
multi_bert-base-cased_binary_complexity_20250409_180139
multi_bert-base-cased_binary_complexity_20250409_185057
multi_bert-base-cased_binary_complexity_20250409_185213
multi_bert-base-cased_binary_complexity_20250409_185333
multi_bert-base-cased_binary_complexity_20250409_234934
multi_bert-base-cased_binary_complexity_20250410_001637
multi_bert-base-cased_binary_complexity_20250410_003117
multi_bert-base-cased_binary_complexity_20250410_004527
multi_bert-base-cased_binary_complexity_20250410_025823
multi_bert-base-cased_binary_complexity_20250410_030623
multi_bert-base-cased_binary_complexity_20250410_031401
multi_bert-base-cased_binary_complexity_20250410_032138
multi_bert-base-cased_binary_complexity_20250410_034203
multi_bert-base-cased_binary_complexity_20250410_034823
multi_bert-base-cased_binary_complexity_20250410_035510
multi_bert-base-cased_binary_complexity_20250410_040140
multi_bert-base-cased_binary_complexity_20250410_174340
multi_bert-base-cased_binary_complexity_20250411_002219
```

multi_bert-base-cased_binary_complexity_20250411_044230
multi_bert-base-cased_binary_complexity_20250411_045829
multi_bert-base-cased_binary_complexity_75th_split_20250411_005437
multi_bert-large-cased_binary_complexity_20250411_002650
multi_bert-large-cased_binary_complexity_20250411_044710
multi_bert-large-cased_binary_complexity_20250411_050144
multi_bert-large-cased_binary_complexity_75th_split_20250411_010152
multi_microsoft
multi_roberta-base_binary_complexity_20250411_002307
multi_roberta-base_binary_complexity_20250411_044250
multi_roberta-base_binary_complexity_20250411_045856
multi_roberta-base_binary_complexity_75th_split_20250411_005524
multi_roberta-large_binary_complexity_20250411_002759
multi_roberta-large_binary_complexity_20250411_044824
multi_roberta-large_binary_complexity_20250411_050222
multi_roberta-large_binary_complexity_75th_split_20250411_010302
multi_xlnet
single_answerdotai
single_bert-base-cased_binary_complexity_20250408_043117
single_bert-base-cased_binary_complexity_20250408_043334
single_bert-base-cased_binary_complexity_20250408_043750
single_bert-base-cased_binary_complexity_20250409_175702
single_bert-base-cased_binary_complexity_20250409_175900
single_bert-base-cased_binary_complexity_20250409_180045
single_bert-base-cased_binary_complexity_20250409_185027
single_bert-base-cased_binary_complexity_20250409_185141
single_bert-base-cased_binary_complexity_20250409_185303
single_bert-base-cased_binary_complexity_20250409_234236
single_bert-base-cased_binary_complexity_20250410_000508
single_bert-base-cased_binary_complexity_20250410_002813
single_bert-base-cased_binary_complexity_20250410_004230
single_bert-base-cased_binary_complexity_20250410_025214
single_bert-base-cased_binary_complexity_20250410_030435
single_bert-base-cased_binary_complexity_20250410_031211
single_bert-base-cased_binary_complexity_20250410_031404
single_bert-base-cased_binary_complexity_20250410_031948
single_bert-base-cased_binary_complexity_20250410_034334
single_bert-base-cased_binary_complexity_20250410_035314
single_bert-base-cased_binary_complexity_20250410_035940
single_bert-base-cased_binary_complexity_20250410_173757
single_bert-base-cased_binary_complexity_20250410_173911
single_bert-base-cased_binary_complexity_20250410_174027
single_bert-base-cased_binary_complexity_20250410_175501
single_bert-base-cased_binary_complexity_20250410_210219
single_bert-base-cased_binary_complexity_20250410_213212
single_bert-base-cased_binary_complexity_20250410_214441
single_bert-base-cased_binary_complexity_20250410_214546
single_bert-base-cased_binary_complexity_20250410_214659

```
single_bert-base-cased_binary_complexity_20250411_044221
single_bert-base-cased_binary_complexity_20250411_044245
single_bert-base-cased_binary_complexity_20250411_120751
single_bert-base-cased_binary_complexity_75th_split_20250411_005451
single_bert-large-cased_binary_complexity_20250410_215725
single_bert-large-cased_binary_complexity_20250410_222431
single_bert-large-cased_binary_complexity_20250411_044617
single_bert-large-cased_binary_complexity_20250411_044715
single_bert-large-cased_binary_complexity_20250411_121110
single_bert-large-cased_binary_complexity_75th_split_20250411_010303
single_microsoft
single_roberta-base_binary_complexity_20250410_212304
single_roberta-base_binary_complexity_20250410_212514
single_roberta-base_binary_complexity_20250410_213732
single_roberta-base_binary_complexity_20250410_214805
single_roberta-base_binary_complexity_20250410_221944
single_roberta-base_binary_complexity_20250411_044307
single_roberta-base_binary_complexity_20250411_044327
single_roberta-base_binary_complexity_20250411_120834
single_roberta-base_binary_complexity_75th_split_20250411_005603
single_roberta-large_binary_complexity_20250410_221054
single_roberta-large_binary_complexity_20250410_222652
single_roberta-large_binary_complexity_20250410_223030
single_roberta-large_binary_complexity_20250410_223320
single_roberta-large_binary_complexity_20250410_223754
single_roberta-large_binary_complexity_20250411_044805
single_roberta-large_binary_complexity_20250411_044938
single_roberta-large_binary_complexity_20250411_121257
single_roberta-large_binary_complexity_75th_split_20250411_010518
single_xlnet
```

[90]:
```python
# Load Model & Tokenizer
# model, tokenizer = get_model_and_tokenizer(named_model) # deprecated argument structure
# model, tokenizer = get_model_and_tokenizer("/content/drive/MyDrive/266-final/models/....") # proposed argument usage for checkpointed models

# for name, param in model.named_parameters():
#     print(name)

model, tokenizer = get_model_and_tokenizer(
    remote_model_name="bert-base-cased",
    local_model_path=None,
    config=custom_config
)

# model, tokenizer = get_model_and_tokenizer(
```

```
#     local_model_path="my_local_bert_path",
#     config=custom_config
# )

print("=============")
print(named_model, ":")
print("=============")
# print(model)
print("=============")
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("=============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```

```
Loading from Hugging Face model: bert-base-cased
```

```
Some weights of BertForSequenceClassification were not initialized from the
model checkpoint at bert-base-cased and are newly initialized:
['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
=============
bert-base-cased :
=============
=============
BertConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
```

```
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}

=============
num_parameters: 108311810
=============
num_trainable_parameters: 108311810
```

Layer Configuration ** Run **

[91]:
```
# Freeze/Unfreeze Layers & Additional Activation Function Configuration

layers_to_unfreeze = [
    # "bert.embeddings.",
    # "bert.encoder.layer.0.",
    # "bert.encoder.layer.1.",
    "bert.encoder.layer.8.",
    "bert.encoder.layer.9.",
    "bert.encoder.layer.10.",
    "bert.encoder.layer.11.",
    "bert.pooler.",
    "classifier.",
]

freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)

for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)

print("\nLayers that are 'True' are trainable. 'False' are frozen.")

print("=============")
print(named_model, ":")
print("=============")
# print(model)
print("=============")
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("=============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```

```
bert.embeddings.word_embeddings.weight requires_grad= False
bert.embeddings.position_embeddings.weight requires_grad= False
bert.embeddings.token_type_embeddings.weight requires_grad= False
bert.embeddings.LayerNorm.weight requires_grad= False
bert.embeddings.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.0.attention.self.query.weight requires_grad= False
bert.encoder.layer.0.attention.self.query.bias requires_grad= False
bert.encoder.layer.0.attention.self.key.weight requires_grad= False
bert.encoder.layer.0.attention.self.key.bias requires_grad= False
bert.encoder.layer.0.attention.self.value.weight requires_grad= False
bert.encoder.layer.0.attention.self.value.bias requires_grad= False
bert.encoder.layer.0.attention.output.dense.weight requires_grad= False
bert.encoder.layer.0.attention.output.dense.bias requires_grad= False
bert.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.0.intermediate.dense.weight requires_grad= False
bert.encoder.layer.0.intermediate.dense.bias requires_grad= False
bert.encoder.layer.0.output.dense.weight requires_grad= False
bert.encoder.layer.0.output.dense.bias requires_grad= False
bert.encoder.layer.0.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.0.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.1.attention.self.query.weight requires_grad= False
bert.encoder.layer.1.attention.self.query.bias requires_grad= False
bert.encoder.layer.1.attention.self.key.weight requires_grad= False
bert.encoder.layer.1.attention.self.key.bias requires_grad= False
bert.encoder.layer.1.attention.self.value.weight requires_grad= False
bert.encoder.layer.1.attention.self.value.bias requires_grad= False
bert.encoder.layer.1.attention.output.dense.weight requires_grad= False
bert.encoder.layer.1.attention.output.dense.bias requires_grad= False
bert.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.1.intermediate.dense.weight requires_grad= False
bert.encoder.layer.1.intermediate.dense.bias requires_grad= False
bert.encoder.layer.1.output.dense.weight requires_grad= False
bert.encoder.layer.1.output.dense.bias requires_grad= False
bert.encoder.layer.1.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.1.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.2.attention.self.query.weight requires_grad= False
bert.encoder.layer.2.attention.self.query.bias requires_grad= False
bert.encoder.layer.2.attention.self.key.weight requires_grad= False
bert.encoder.layer.2.attention.self.key.bias requires_grad= False
bert.encoder.layer.2.attention.self.value.weight requires_grad= False
bert.encoder.layer.2.attention.self.value.bias requires_grad= False
bert.encoder.layer.2.attention.output.dense.weight requires_grad= False
bert.encoder.layer.2.attention.output.dense.bias requires_grad= False
bert.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.2.intermediate.dense.weight requires_grad= False
bert.encoder.layer.2.intermediate.dense.bias requires_grad= False
bert.encoder.layer.2.output.dense.weight requires_grad= False
bert.encoder.layer.2.output.dense.bias requires_grad= False
bert.encoder.layer.2.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.2.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.3.attention.self.query.weight requires_grad= False
bert.encoder.layer.3.attention.self.query.bias requires_grad= False
bert.encoder.layer.3.attention.self.key.weight requires_grad= False
bert.encoder.layer.3.attention.self.key.bias requires_grad= False
bert.encoder.layer.3.attention.self.value.weight requires_grad= False
bert.encoder.layer.3.attention.self.value.bias requires_grad= False
bert.encoder.layer.3.attention.output.dense.weight requires_grad= False
bert.encoder.layer.3.attention.output.dense.bias requires_grad= False
bert.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.3.intermediate.dense.weight requires_grad= False
bert.encoder.layer.3.intermediate.dense.bias requires_grad= False
bert.encoder.layer.3.output.dense.weight requires_grad= False
bert.encoder.layer.3.output.dense.bias requires_grad= False
bert.encoder.layer.3.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.3.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.4.attention.self.query.weight requires_grad= False
bert.encoder.layer.4.attention.self.query.bias requires_grad= False
bert.encoder.layer.4.attention.self.key.weight requires_grad= False
bert.encoder.layer.4.attention.self.key.bias requires_grad= False
bert.encoder.layer.4.attention.self.value.weight requires_grad= False
bert.encoder.layer.4.attention.self.value.bias requires_grad= False
bert.encoder.layer.4.attention.output.dense.weight requires_grad= False
bert.encoder.layer.4.attention.output.dense.bias requires_grad= False
bert.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.4.intermediate.dense.weight requires_grad= False
bert.encoder.layer.4.intermediate.dense.bias requires_grad= False
bert.encoder.layer.4.output.dense.weight requires_grad= False
bert.encoder.layer.4.output.dense.bias requires_grad= False
bert.encoder.layer.4.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.4.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.5.attention.self.query.weight requires_grad= False
bert.encoder.layer.5.attention.self.query.bias requires_grad= False
bert.encoder.layer.5.attention.self.key.weight requires_grad= False
bert.encoder.layer.5.attention.self.key.bias requires_grad= False
bert.encoder.layer.5.attention.self.value.weight requires_grad= False
bert.encoder.layer.5.attention.self.value.bias requires_grad= False
bert.encoder.layer.5.attention.output.dense.weight requires_grad= False
bert.encoder.layer.5.attention.output.dense.bias requires_grad= False
bert.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.5.intermediate.dense.weight requires_grad= False
bert.encoder.layer.5.intermediate.dense.bias requires_grad= False
bert.encoder.layer.5.output.dense.weight requires_grad= False
bert.encoder.layer.5.output.dense.bias requires_grad= False
bert.encoder.layer.5.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.5.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.6.attention.self.query.weight requires_grad= False
bert.encoder.layer.6.attention.self.query.bias requires_grad= False
bert.encoder.layer.6.attention.self.key.weight requires_grad= False
bert.encoder.layer.6.attention.self.key.bias requires_grad= False
bert.encoder.layer.6.attention.self.value.weight requires_grad= False
bert.encoder.layer.6.attention.self.value.bias requires_grad= False
bert.encoder.layer.6.attention.output.dense.weight requires_grad= False
bert.encoder.layer.6.attention.output.dense.bias requires_grad= False
bert.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.6.intermediate.dense.weight requires_grad= False
bert.encoder.layer.6.intermediate.dense.bias requires_grad= False
bert.encoder.layer.6.output.dense.weight requires_grad= False
bert.encoder.layer.6.output.dense.bias requires_grad= False
bert.encoder.layer.6.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.6.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.7.attention.self.query.weight requires_grad= False
bert.encoder.layer.7.attention.self.query.bias requires_grad= False
bert.encoder.layer.7.attention.self.key.weight requires_grad= False
bert.encoder.layer.7.attention.self.key.bias requires_grad= False
bert.encoder.layer.7.attention.self.value.weight requires_grad= False
bert.encoder.layer.7.attention.self.value.bias requires_grad= False
bert.encoder.layer.7.attention.output.dense.weight requires_grad= False
bert.encoder.layer.7.attention.output.dense.bias requires_grad= False
bert.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.7.intermediate.dense.weight requires_grad= False
bert.encoder.layer.7.intermediate.dense.bias requires_grad= False
bert.encoder.layer.7.output.dense.weight requires_grad= False
bert.encoder.layer.7.output.dense.bias requires_grad= False
bert.encoder.layer.7.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.7.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.8.attention.self.query.weight requires_grad= True
bert.encoder.layer.8.attention.self.query.bias requires_grad= True
bert.encoder.layer.8.attention.self.key.weight requires_grad= True
bert.encoder.layer.8.attention.self.key.bias requires_grad= True
bert.encoder.layer.8.attention.self.value.weight requires_grad= True
bert.encoder.layer.8.attention.self.value.bias requires_grad= True
bert.encoder.layer.8.attention.output.dense.weight requires_grad= True
bert.encoder.layer.8.attention.output.dense.bias requires_grad= True
bert.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.8.intermediate.dense.weight requires_grad= True
bert.encoder.layer.8.intermediate.dense.bias requires_grad= True
bert.encoder.layer.8.output.dense.weight requires_grad= True
bert.encoder.layer.8.output.dense.bias requires_grad= True
bert.encoder.layer.8.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.8.output.LayerNorm.bias requires_grad= True
```

```
bert.encoder.layer.9.attention.self.query.weight requires_grad= True
bert.encoder.layer.9.attention.self.query.bias requires_grad= True
bert.encoder.layer.9.attention.self.key.weight requires_grad= True
bert.encoder.layer.9.attention.self.key.bias requires_grad= True
bert.encoder.layer.9.attention.self.value.weight requires_grad= True
bert.encoder.layer.9.attention.self.value.bias requires_grad= True
bert.encoder.layer.9.attention.output.dense.weight requires_grad= True
bert.encoder.layer.9.attention.output.dense.bias requires_grad= True
bert.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.9.intermediate.dense.weight requires_grad= True
bert.encoder.layer.9.intermediate.dense.bias requires_grad= True
bert.encoder.layer.9.output.dense.weight requires_grad= True
bert.encoder.layer.9.output.dense.bias requires_grad= True
bert.encoder.layer.9.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.9.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.10.attention.self.query.weight requires_grad= True
bert.encoder.layer.10.attention.self.query.bias requires_grad= True
bert.encoder.layer.10.attention.self.key.weight requires_grad= True
bert.encoder.layer.10.attention.self.key.bias requires_grad= True
bert.encoder.layer.10.attention.self.value.weight requires_grad= True
bert.encoder.layer.10.attention.self.value.bias requires_grad= True
bert.encoder.layer.10.attention.output.dense.weight requires_grad= True
bert.encoder.layer.10.attention.output.dense.bias requires_grad= True
bert.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.10.intermediate.dense.weight requires_grad= True
bert.encoder.layer.10.intermediate.dense.bias requires_grad= True
bert.encoder.layer.10.output.dense.weight requires_grad= True
bert.encoder.layer.10.output.dense.bias requires_grad= True
bert.encoder.layer.10.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.10.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.11.attention.self.query.weight requires_grad= True
bert.encoder.layer.11.attention.self.query.bias requires_grad= True
bert.encoder.layer.11.attention.self.key.weight requires_grad= True
bert.encoder.layer.11.attention.self.key.bias requires_grad= True
bert.encoder.layer.11.attention.self.value.weight requires_grad= True
bert.encoder.layer.11.attention.self.value.bias requires_grad= True
bert.encoder.layer.11.attention.output.dense.weight requires_grad= True
bert.encoder.layer.11.attention.output.dense.bias requires_grad= True
bert.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.11.intermediate.dense.weight requires_grad= True
bert.encoder.layer.11.intermediate.dense.bias requires_grad= True
bert.encoder.layer.11.output.dense.weight requires_grad= True
bert.encoder.layer.11.output.dense.bias requires_grad= True
bert.encoder.layer.11.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.11.output.LayerNorm.bias requires_grad= True
```

```
bert.pooler.dense.weight requires_grad= True
bert.pooler.dense.bias requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True

Layers that are 'True' are trainable. 'False' are frozen.
=============
bert-base-cased :
=============
=============
BertConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}

=============
num_parameters: 108311810
=============
num_trainable_parameters: 28943618
```

**Dataset Preparation ** Run **

[92]:
```
# Tokenize & Prepare Datasets

train_data_hf = prepare_dataset(
    df_train,
```

```
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max
)

val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max
)

test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max

)

print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/421 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/917 [00:00<?, ? examples/s]
```

```
Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([  101,  1252,  1106,  1103,  3824,
1104, 19892, 11220,  1324,  1119,
         1522,  3839,   117,  1272,  1103,  1555,  1104,  1103, 11563,  5609,
         1106,  1172,   132,  1152,  2446,  1122,  1113,  1147,  3221,   119,
          102,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
```

```
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0]), 'attention_mask': tensor([1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])}
```

0.2.3 snc regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[ ]:
```
# Define Experiment Parameters
named_model = "bert-base-cased"
# named_model = "roberta-base"
# named_model = "bert-large"
# named_model = "roberta-large"
# named_model = "" # modern bert
############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
num_epochs = 1
##########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
############
y_col = "binary_complexity"
```

```
# y_col = "complexity"
#############
x_task = "single"
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
########################################
# custom_config = BertConfig.from_pretrained("bert-base-cased")
# custom_config.hidden_act = "gelu"  # alts: "relu" "silu"
# custom_config.attention_probs_dropout_prob = 0.1
# custom_config.hidden_dropout_prob = 0.1
# custom_config.gradient_checkpointing = False
########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="bert-base-cased",
    local_model_path=None,
    config=None)
#############
```

```
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
#     local_model_path="...CONFIGURE_PATH...",
#     config=custom_config)
print("=============")
print(named_model, ":")
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters at load:", model.
 ↪num_parameters(only_trainable=True))
print("=============")
print("model lineage:", MODEL_LINEAGE)
print("=============")
########################################
layers_to_unfreeze = [
    # "bert.embeddings.",
    # "bert.encoder.layer.0.",
    # "bert.encoder.layer.1.",
    # "bert.encoder.layer.8.",
    # "bert.encoder.layer.9.",
    # "bert.encoder.layer.10.",
    "bert.encoder.layer.11.",
    "bert.pooler.",
    "classifier.",
]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("=============")
########################################
print("Experiment configuration used with this experiment:")
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]
Map:   0%|          | 0/421 [00:00<?, ? examples/s]
Map:   0%|          | 0/917 [00:00<?, ? examples/s]
```

```
Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([  101,  1252,  1106,  1103,  3824,
1104, 19892, 11220,  1324,  1119,
         1522,  3839,   117,  1272,  1103,  1555,  1104,  1103, 11563,  5609,
         1106,  1172,   132,  1152,  2446,  1122,  1113,  1147,  3221,   119,
          102,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0]),
'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])}
Loading from Hugging Face model: bert-base-cased

Some weights of BertForSequenceClassification were not initialized from the
model checkpoint at bert-base-cased and are newly initialized:
['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

=============
bert-base-cased :
=============
num_parameters: 108311810
num_trainable_parameters at load: 108311810
=============
model lineage: {'type': 'huggingface_hub', 'path': 'bert-base-cased',
'timestamp': '2025-04-11 12:07:29'}
=============
BertConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
```

```
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}

=============
num_parameters: 108311810
num_trainable_parameters: 7680002
=============
Experiment configuration used with this experiment:
model used: bert-base-cased
learning rate used: 5e-06
number of epochs: 1
maximum sequence length: 128
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
```

[ ]:
```
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
    val_dataset = val_data_hf,
    output_dir = dir_results,
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-20-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

```
<IPython.core.display.HTML object>
```

```
Downloading builder script:   0%|          | 0.00/4.20k [00:00<?, ?B/s]
```

```
Downloading builder script:   0%|          | 0.00/7.56k [00:00<?, ?B/s]
```

```
Downloading builder script:   0%|          | 0.00/7.38k [00:00<?, ?B/s]
```

```
Downloading builder script:   0%|          | 0.00/6.79k [00:00<?, ?B/s]
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 0.6972283720970154, 'eval_accuracy':
0.47268408551068886, 'eval_precision': 0.39285714285714285, 'eval_recall':
0.2864583333333333, 'eval_f1': 0.3313253012048193, 'eval_runtime': 1.7194,
'eval_samples_per_second': 244.857, 'eval_steps_per_second': 2.326, 'epoch':
1.0}
Test metrics: {'eval_loss': 0.6932057738304138, 'eval_accuracy':
0.5212649945474372, 'eval_precision': 0.5045871559633027, 'eval_recall':
0.2494331065759637, 'eval_f1': 0.3338391502276176, 'eval_runtime': 2.3778,
'eval_samples_per_second': 385.656, 'eval_steps_per_second': 3.364, 'epoch':
1.0}
```

[ ]:
```python
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
  ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
```

```
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to: /content/drive/MyDrive/266-final/models/single_bert-
base-cased_binary_complexity_20250411_120751

<IPython.core.display.HTML object>

EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```

0.2.4 snc roberta-base regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[ ]:
```
# Define Experiment Parameters
# named_model = "bert-base-cased"
named_model = "roberta-base"
# named_model = "bert-large"
# named_model = "roberta-large"
# named_model = "" # modern bert
#############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
num_epochs = 1
#########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
#############
y_col = "binary_complexity"
# y_col = "complexity"
#############
x_task = "single"
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
```

```
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
########################################
# custom_config = BertConfig.from_pretrained("roberta-base")
# custom_config.hidden_act = "gelu"  # alts: "relu" "silu"
# custom_config.attention_probs_dropout_prob = 0.1
# custom_config.hidden_dropout_prob = 0.1
# custom_config.gradient_checkpointing = False
########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="roberta-base",
    local_model_path=None,
    config=None)
############
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
#     local_model_path="...CONFIGURE_PATH...",
#     config=custom_config)
print("=============")
print(named_model, ":")
```

```
print("==============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters at load:", model.
  ↪num_parameters(only_trainable=True))
print("==============")
print("model lineage:", MODEL_LINEAGE)
print("==============")
########################################
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]

Map:   0%|          | 0/421 [00:00<?, ? examples/s]

Map:   0%|          | 0/917 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([  101,  1252,  1106,  1103,  3824,
1104, 19892, 11220,  1324,  1119,
         1522,  3839,   117,  1272,  1103,  1555,  1104,  1103, 11563,  5609,
         1106,  1172,   132,  1152,  2446,  1122,  1113,  1147,  3221,   119,
          102,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0]),
'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])}
Loading from Hugging Face model: roberta-base

tokenizer_config.json:   0%|          | 0.00/25.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/481 [00:00<?, ?B/s]

vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]

merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.36M [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/499M [00:00<?, ?B/s]

Some weights of RobertaForSequenceClassification were not initialized from the
model checkpoint at roberta-base and are newly initialized:
```

```
['classifier.dense.bias', 'classifier.dense.weight', 'classifier.out_proj.bias',
'classifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

=============
roberta-base :
=============
num_parameters: 124647170
num_trainable_parameters at load: 124647170
=============
model lineage: {'type': 'huggingface_hub', 'path': 'roberta-base', 'timestamp':
'2025-04-11 12:08:14'}
=============
```

[ ]:
```
print(model)
```

```
RobertaForSequenceClassification(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(50265, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0-11): 12 x RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSdpaSelfAttention(
              (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)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
```

```
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (classifier): RobertaClassificationHead(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
    (out_proj): Linear(in_features=768, out_features=2, bias=True)
  )
)
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
roberta.embeddings.word_embeddings.weight requires_grad= True
roberta.embeddings.position_embeddings.weight requires_grad= True
roberta.embeddings.token_type_embeddings.weight requires_grad= True
roberta.embeddings.LayerNorm.weight requires_grad= True
roberta.embeddings.LayerNorm.bias requires_grad= True
roberta.encoder.layer.0.attention.self.query.weight requires_grad= True
roberta.encoder.layer.0.attention.self.query.bias requires_grad= True
roberta.encoder.layer.0.attention.self.key.weight requires_grad= True
roberta.encoder.layer.0.attention.self.key.bias requires_grad= True
roberta.encoder.layer.0.attention.self.value.weight requires_grad= True
roberta.encoder.layer.0.attention.self.value.bias requires_grad= True
roberta.encoder.layer.0.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.0.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.0.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.0.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.0.output.dense.weight requires_grad= True
roberta.encoder.layer.0.output.dense.bias requires_grad= True
roberta.encoder.layer.0.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.0.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.1.attention.self.query.weight requires_grad= True
roberta.encoder.layer.1.attention.self.query.bias requires_grad= True
roberta.encoder.layer.1.attention.self.key.weight requires_grad= True
roberta.encoder.layer.1.attention.self.key.bias requires_grad= True
roberta.encoder.layer.1.attention.self.value.weight requires_grad= True
roberta.encoder.layer.1.attention.self.value.bias requires_grad= True
roberta.encoder.layer.1.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.1.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.1.intermediate.dense.weight requires_grad= True
```

```
roberta.encoder.layer.1.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.1.output.dense.weight requires_grad= True
roberta.encoder.layer.1.output.dense.bias requires_grad= True
roberta.encoder.layer.1.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.1.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.2.attention.self.query.weight requires_grad= True
roberta.encoder.layer.2.attention.self.query.bias requires_grad= True
roberta.encoder.layer.2.attention.self.key.weight requires_grad= True
roberta.encoder.layer.2.attention.self.key.bias requires_grad= True
roberta.encoder.layer.2.attention.self.value.weight requires_grad= True
roberta.encoder.layer.2.attention.self.value.bias requires_grad= True
roberta.encoder.layer.2.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.2.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.2.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.2.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.2.output.dense.weight requires_grad= True
roberta.encoder.layer.2.output.dense.bias requires_grad= True
roberta.encoder.layer.2.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.2.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.3.attention.self.query.weight requires_grad= True
roberta.encoder.layer.3.attention.self.query.bias requires_grad= True
roberta.encoder.layer.3.attention.self.key.weight requires_grad= True
roberta.encoder.layer.3.attention.self.key.bias requires_grad= True
roberta.encoder.layer.3.attention.self.value.weight requires_grad= True
roberta.encoder.layer.3.attention.self.value.bias requires_grad= True
roberta.encoder.layer.3.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.3.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.3.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.3.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.3.output.dense.weight requires_grad= True
roberta.encoder.layer.3.output.dense.bias requires_grad= True
roberta.encoder.layer.3.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.3.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.4.attention.self.query.weight requires_grad= True
roberta.encoder.layer.4.attention.self.query.bias requires_grad= True
roberta.encoder.layer.4.attention.self.key.weight requires_grad= True
roberta.encoder.layer.4.attention.self.key.bias requires_grad= True
roberta.encoder.layer.4.attention.self.value.weight requires_grad= True
roberta.encoder.layer.4.attention.self.value.bias requires_grad= True
roberta.encoder.layer.4.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.4.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.4.intermediate.dense.weight requires_grad= True
```

roberta.encoder.layer.4.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.4.output.dense.weight requires_grad= True
roberta.encoder.layer.4.output.dense.bias requires_grad= True
roberta.encoder.layer.4.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.4.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.5.attention.self.query.weight requires_grad= True
roberta.encoder.layer.5.attention.self.query.bias requires_grad= True
roberta.encoder.layer.5.attention.self.key.weight requires_grad= True
roberta.encoder.layer.5.attention.self.key.bias requires_grad= True
roberta.encoder.layer.5.attention.self.value.weight requires_grad= True
roberta.encoder.layer.5.attention.self.value.bias requires_grad= True
roberta.encoder.layer.5.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.5.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.5.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.5.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.5.output.dense.weight requires_grad= True
roberta.encoder.layer.5.output.dense.bias requires_grad= True
roberta.encoder.layer.5.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.5.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.6.attention.self.query.weight requires_grad= True
roberta.encoder.layer.6.attention.self.query.bias requires_grad= True
roberta.encoder.layer.6.attention.self.key.weight requires_grad= True
roberta.encoder.layer.6.attention.self.key.bias requires_grad= True
roberta.encoder.layer.6.attention.self.value.weight requires_grad= True
roberta.encoder.layer.6.attention.self.value.bias requires_grad= True
roberta.encoder.layer.6.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.6.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.6.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.6.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.6.output.dense.weight requires_grad= True
roberta.encoder.layer.6.output.dense.bias requires_grad= True
roberta.encoder.layer.6.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.6.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.7.attention.self.query.weight requires_grad= True
roberta.encoder.layer.7.attention.self.query.bias requires_grad= True
roberta.encoder.layer.7.attention.self.key.weight requires_grad= True
roberta.encoder.layer.7.attention.self.key.bias requires_grad= True
roberta.encoder.layer.7.attention.self.value.weight requires_grad= True
roberta.encoder.layer.7.attention.self.value.bias requires_grad= True
roberta.encoder.layer.7.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.7.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.7.intermediate.dense.weight requires_grad= True

roberta.encoder.layer.7.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.7.output.dense.weight requires_grad= True
roberta.encoder.layer.7.output.dense.bias requires_grad= True
roberta.encoder.layer.7.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.7.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.8.attention.self.query.weight requires_grad= True
roberta.encoder.layer.8.attention.self.query.bias requires_grad= True
roberta.encoder.layer.8.attention.self.key.weight requires_grad= True
roberta.encoder.layer.8.attention.self.key.bias requires_grad= True
roberta.encoder.layer.8.attention.self.value.weight requires_grad= True
roberta.encoder.layer.8.attention.self.value.bias requires_grad= True
roberta.encoder.layer.8.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.8.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.8.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.8.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.8.output.dense.weight requires_grad= True
roberta.encoder.layer.8.output.dense.bias requires_grad= True
roberta.encoder.layer.8.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.8.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.9.attention.self.query.weight requires_grad= True
roberta.encoder.layer.9.attention.self.query.bias requires_grad= True
roberta.encoder.layer.9.attention.self.key.weight requires_grad= True
roberta.encoder.layer.9.attention.self.key.bias requires_grad= True
roberta.encoder.layer.9.attention.self.value.weight requires_grad= True
roberta.encoder.layer.9.attention.self.value.bias requires_grad= True
roberta.encoder.layer.9.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.9.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.9.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.9.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.9.output.dense.weight requires_grad= True
roberta.encoder.layer.9.output.dense.bias requires_grad= True
roberta.encoder.layer.9.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.9.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.10.attention.self.query.weight requires_grad= True
roberta.encoder.layer.10.attention.self.query.bias requires_grad= True
roberta.encoder.layer.10.attention.self.key.weight requires_grad= True
roberta.encoder.layer.10.attention.self.key.bias requires_grad= True
roberta.encoder.layer.10.attention.self.value.weight requires_grad= True
roberta.encoder.layer.10.attention.self.value.bias requires_grad= True
roberta.encoder.layer.10.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.10.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.10.intermediate.dense.weight requires_grad= True

```
roberta.encoder.layer.10.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.10.output.dense.weight requires_grad= True
roberta.encoder.layer.10.output.dense.bias requires_grad= True
roberta.encoder.layer.10.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.10.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.11.attention.self.query.weight requires_grad= True
roberta.encoder.layer.11.attention.self.query.bias requires_grad= True
roberta.encoder.layer.11.attention.self.key.weight requires_grad= True
roberta.encoder.layer.11.attention.self.key.bias requires_grad= True
roberta.encoder.layer.11.attention.self.value.weight requires_grad= True
roberta.encoder.layer.11.attention.self.value.bias requires_grad= True
roberta.encoder.layer.11.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.11.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.11.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.11.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.11.output.dense.weight requires_grad= True
roberta.encoder.layer.11.output.dense.bias requires_grad= True
roberta.encoder.layer.11.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.11.output.LayerNorm.bias requires_grad= True
classifier.dense.weight requires_grad= True
classifier.dense.bias requires_grad= True
classifier.out_proj.weight requires_grad= True
classifier.out_proj.bias requires_grad= True
```

[ ]:
```python
# Inspect the attention_mask tensor for the first few samples
for i in range(5):
    print(train_data_hf[i]['attention_mask'])
```

```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
```

```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
```

[ ]:
```python
layers_to_unfreeze = [
    "roberta.encoder.layer.11.attention.self.query.weight",
    "roberta.encoder.layer.11.attention.self.query.bias",
    "roberta.encoder.layer.11.attention.self.key.weight",
    "roberta.encoder.layer.11.attention.self.key.bias",
    "roberta.encoder.layer.11.attention.self.value.weight",
    "roberta.encoder.layer.11.attention.self.value.bias",
    "roberta.encoder.layer.11.attention.output.dense.weight",
    "roberta.encoder.layer.11.attention.output.dense.bias",
    "roberta.encoder.layer.11.attention.output.LayerNorm.weight",
    "roberta.encoder.layer.11.attention.output.LayerNorm.bias",
    "roberta.encoder.layer.11.intermediate.dense.weight",
    "roberta.encoder.layer.11.intermediate.dense.bias",
    "roberta.encoder.layer.11.output.dense.weight",
    "roberta.encoder.layer.11.output.dense.bias",
    "roberta.encoder.layer.11.output.LayerNorm.weight",
    "roberta.encoder.layer.11.output.LayerNorm.bias",
    "classifier.dense.weight",
    "classifier.dense.bias",
    "classifier.out_proj.weight",
    "classifier.out_proj.bias"
]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("=============")
########################################
print("Experiment configuration used with this experiment:")
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
```

```
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
```

```
RobertaConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "RobertaForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-05,
  "max_position_embeddings": 514,
  "model_type": "roberta",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "type_vocab_size": 1,
  "use_cache": true,
  "vocab_size": 50265
}

=============
num_parameters: 124647170
num_trainable_parameters: 7680002
=============
Experiment configuration used with this experiment:
model used: roberta-base
learning rate used: 5e-06
number of epochs: 1
maximum sequence length: 128
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
```

```
[ ]: for name, param in model.named_parameters():
         print(name, "requires_grad=", param.requires_grad)
```

```
roberta.embeddings.word_embeddings.weight requires_grad= False
roberta.embeddings.position_embeddings.weight requires_grad= False
roberta.embeddings.token_type_embeddings.weight requires_grad= False
roberta.embeddings.LayerNorm.weight requires_grad= False
roberta.embeddings.LayerNorm.bias requires_grad= False
roberta.encoder.layer.0.attention.self.query.weight requires_grad= False
roberta.encoder.layer.0.attention.self.query.bias requires_grad= False
roberta.encoder.layer.0.attention.self.key.weight requires_grad= False
roberta.encoder.layer.0.attention.self.key.bias requires_grad= False
roberta.encoder.layer.0.attention.self.value.weight requires_grad= False
roberta.encoder.layer.0.attention.self.value.bias requires_grad= False
roberta.encoder.layer.0.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.0.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.0.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.0.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.0.output.dense.weight requires_grad= False
roberta.encoder.layer.0.output.dense.bias requires_grad= False
roberta.encoder.layer.0.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.0.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.1.attention.self.query.weight requires_grad= False
roberta.encoder.layer.1.attention.self.query.bias requires_grad= False
roberta.encoder.layer.1.attention.self.key.weight requires_grad= False
roberta.encoder.layer.1.attention.self.key.bias requires_grad= False
roberta.encoder.layer.1.attention.self.value.weight requires_grad= False
roberta.encoder.layer.1.attention.self.value.bias requires_grad= False
roberta.encoder.layer.1.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.1.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.1.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.1.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.1.output.dense.weight requires_grad= False
roberta.encoder.layer.1.output.dense.bias requires_grad= False
roberta.encoder.layer.1.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.1.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.2.attention.self.query.weight requires_grad= False
roberta.encoder.layer.2.attention.self.query.bias requires_grad= False
roberta.encoder.layer.2.attention.self.key.weight requires_grad= False
roberta.encoder.layer.2.attention.self.key.bias requires_grad= False
roberta.encoder.layer.2.attention.self.value.weight requires_grad= False
roberta.encoder.layer.2.attention.self.value.bias requires_grad= False
roberta.encoder.layer.2.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.2.attention.output.dense.bias requires_grad= False
```

roberta.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.2.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.2.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.2.output.dense.weight requires_grad= False
roberta.encoder.layer.2.output.dense.bias requires_grad= False
roberta.encoder.layer.2.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.2.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.3.attention.self.query.weight requires_grad= False
roberta.encoder.layer.3.attention.self.query.bias requires_grad= False
roberta.encoder.layer.3.attention.self.key.weight requires_grad= False
roberta.encoder.layer.3.attention.self.key.bias requires_grad= False
roberta.encoder.layer.3.attention.self.value.weight requires_grad= False
roberta.encoder.layer.3.attention.self.value.bias requires_grad= False
roberta.encoder.layer.3.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.3.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.3.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.3.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.3.output.dense.weight requires_grad= False
roberta.encoder.layer.3.output.dense.bias requires_grad= False
roberta.encoder.layer.3.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.3.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.4.attention.self.query.weight requires_grad= False
roberta.encoder.layer.4.attention.self.query.bias requires_grad= False
roberta.encoder.layer.4.attention.self.key.weight requires_grad= False
roberta.encoder.layer.4.attention.self.key.bias requires_grad= False
roberta.encoder.layer.4.attention.self.value.weight requires_grad= False
roberta.encoder.layer.4.attention.self.value.bias requires_grad= False
roberta.encoder.layer.4.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.4.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.4.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.4.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.4.output.dense.weight requires_grad= False
roberta.encoder.layer.4.output.dense.bias requires_grad= False
roberta.encoder.layer.4.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.4.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.5.attention.self.query.weight requires_grad= False
roberta.encoder.layer.5.attention.self.query.bias requires_grad= False
roberta.encoder.layer.5.attention.self.key.weight requires_grad= False
roberta.encoder.layer.5.attention.self.key.bias requires_grad= False
roberta.encoder.layer.5.attention.self.value.weight requires_grad= False
roberta.encoder.layer.5.attention.self.value.bias requires_grad= False
roberta.encoder.layer.5.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.5.attention.output.dense.bias requires_grad= False

roberta.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.5.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.5.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.5.output.dense.weight requires_grad= False
roberta.encoder.layer.5.output.dense.bias requires_grad= False
roberta.encoder.layer.5.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.5.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.6.attention.self.query.weight requires_grad= False
roberta.encoder.layer.6.attention.self.query.bias requires_grad= False
roberta.encoder.layer.6.attention.self.key.weight requires_grad= False
roberta.encoder.layer.6.attention.self.key.bias requires_grad= False
roberta.encoder.layer.6.attention.self.value.weight requires_grad= False
roberta.encoder.layer.6.attention.self.value.bias requires_grad= False
roberta.encoder.layer.6.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.6.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.6.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.6.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.6.output.dense.weight requires_grad= False
roberta.encoder.layer.6.output.dense.bias requires_grad= False
roberta.encoder.layer.6.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.6.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.7.attention.self.query.weight requires_grad= False
roberta.encoder.layer.7.attention.self.query.bias requires_grad= False
roberta.encoder.layer.7.attention.self.key.weight requires_grad= False
roberta.encoder.layer.7.attention.self.key.bias requires_grad= False
roberta.encoder.layer.7.attention.self.value.weight requires_grad= False
roberta.encoder.layer.7.attention.self.value.bias requires_grad= False
roberta.encoder.layer.7.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.7.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.7.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.7.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.7.output.dense.weight requires_grad= False
roberta.encoder.layer.7.output.dense.bias requires_grad= False
roberta.encoder.layer.7.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.7.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.8.attention.self.query.weight requires_grad= False
roberta.encoder.layer.8.attention.self.query.bias requires_grad= False
roberta.encoder.layer.8.attention.self.key.weight requires_grad= False
roberta.encoder.layer.8.attention.self.key.bias requires_grad= False
roberta.encoder.layer.8.attention.self.value.weight requires_grad= False
roberta.encoder.layer.8.attention.self.value.bias requires_grad= False
roberta.encoder.layer.8.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.8.attention.output.dense.bias requires_grad= False

roberta.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.8.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.8.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.8.output.dense.weight requires_grad= False
roberta.encoder.layer.8.output.dense.bias requires_grad= False
roberta.encoder.layer.8.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.8.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.9.attention.self.query.weight requires_grad= False
roberta.encoder.layer.9.attention.self.query.bias requires_grad= False
roberta.encoder.layer.9.attention.self.key.weight requires_grad= False
roberta.encoder.layer.9.attention.self.key.bias requires_grad= False
roberta.encoder.layer.9.attention.self.value.weight requires_grad= False
roberta.encoder.layer.9.attention.self.value.bias requires_grad= False
roberta.encoder.layer.9.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.9.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.9.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.9.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.9.output.dense.weight requires_grad= False
roberta.encoder.layer.9.output.dense.bias requires_grad= False
roberta.encoder.layer.9.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.9.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.10.attention.self.query.weight requires_grad= False
roberta.encoder.layer.10.attention.self.query.bias requires_grad= False
roberta.encoder.layer.10.attention.self.key.weight requires_grad= False
roberta.encoder.layer.10.attention.self.key.bias requires_grad= False
roberta.encoder.layer.10.attention.self.value.weight requires_grad= False
roberta.encoder.layer.10.attention.self.value.bias requires_grad= False
roberta.encoder.layer.10.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.10.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.10.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.10.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.10.output.dense.weight requires_grad= False
roberta.encoder.layer.10.output.dense.bias requires_grad= False
roberta.encoder.layer.10.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.10.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.11.attention.self.query.weight requires_grad= True
roberta.encoder.layer.11.attention.self.query.bias requires_grad= True
roberta.encoder.layer.11.attention.self.key.weight requires_grad= True
roberta.encoder.layer.11.attention.self.key.bias requires_grad= True
roberta.encoder.layer.11.attention.self.value.weight requires_grad= True
roberta.encoder.layer.11.attention.self.value.bias requires_grad= True
roberta.encoder.layer.11.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.11.attention.output.dense.bias requires_grad= True

```
roberta.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.11.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.11.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.11.output.dense.weight requires_grad= True
roberta.encoder.layer.11.output.dense.bias requires_grad= True
roberta.encoder.layer.11.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.11.output.LayerNorm.bias requires_grad= True
classifier.dense.weight requires_grad= True
classifier.dense.bias requires_grad= True
classifier.out_proj.weight requires_grad= True
classifier.out_proj.bias requires_grad= True
```

[ ]:
```python
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
    val_dataset = val_data_hf,
    output_dir = dir_results,
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-20-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 0.6934652328491211, 'eval_accuracy':
0.47743467933491684, 'eval_precision': 0.46296296296296297, 'eval_recall':
0.9114583333333334, 'eval_f1': 0.6140350877192983, 'eval_runtime': 2.0942,
'eval_samples_per_second': 201.03, 'eval_steps_per_second': 1.91, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.6933342218399048, 'eval_accuracy':
0.4841875681570338, 'eval_precision': 0.4805352798053528, 'eval_recall':
0.8956916099773242, 'eval_f1': 0.6254948535233571, 'eval_runtime': 2.5107,
'eval_samples_per_second': 365.23, 'eval_steps_per_second': 3.186, 'epoch': 1.0}
```

[ ]:
```
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
 ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to:
/content/drive/MyDrive/266-final/models/single_roberta-
base_binary_complexity_20250411_120834

<IPython.core.display.HTML object>

EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```

0.2.5 snc bert-large-cased regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[ ]:
```
# Define Experiment Parameters
# named_model = "bert-base-cased"
# named_model = "roberta-base"
named_model = "bert-large-cased"
# named_model = "roberta-large"
# named_model = "" # modern bert
```

```
############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
num_epochs = 1
#########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
############
y_col = "binary_complexity"
# y_col = "complexity"
############
x_task = "single"
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
#########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
```

```python
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
##########################################
# custom_config = BertConfig.from_pretrained("roberta-base")
# custom_config.hidden_act = "gelu"  # alts: "relu" "silu"
# custom_config.attention_probs_dropout_prob = 0.1
# custom_config.hidden_dropout_prob = 0.1
# custom_config.gradient_checkpointing = False
##########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="bert-large-cased",
    local_model_path=None,
    config=None)
############
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
#     local_model_path="...CONFIGURE_PATH...",
#     config=custom_config)
print("=============")
print(named_model, ":")
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters at load:", model.
 ↪num_parameters(only_trainable=True))
print("=============")
print("model lineage:", MODEL_LINEAGE)
print("=============")
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]

Map:   0%|          | 0/421 [00:00<?, ? examples/s]

Map:   0%|          | 0/917 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([  101,  1252,  1106,  1103,  3824,
1104, 19892, 11220,  1324,  1119,
         1522,  3839,   117,  1272,  1103,  1555,  1104,  1103, 11563,  5609,
         1106,  1172,   132,  1152,  2446,  1122,  1113,  1147,  3221,   119,
          102,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0]),
```

```
'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])}
Loading from Hugging Face model: bert-large-cased

Some weights of BertForSequenceClassification were not initialized from the
model checkpoint at bert-large-cased and are newly initialized:
['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

=============
bert-large-cased :
=============
num_parameters: 333581314
num_trainable_parameters at load: 333581314
=============
model lineage: {'type': 'huggingface_hub', 'path': 'bert-large-cased',
'timestamp': '2025-04-11 12:10:19'}
=============
```

[ ]:
```
print(model)
```

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(28996, 1024, padding_idx=0)
      (position_embeddings): Embedding(512, 1024)
      (token_type_embeddings): Embedding(2, 1024)
      (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-23): 24 x BertLayer(
          (attention): BertAttention(
            (self): BertSdpaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-12,
```

```
elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=1024, out_features=1024, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=1024, out_features=2, bias=True)
)
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
bert.embeddings.word_embeddings.weight requires_grad= True
bert.embeddings.position_embeddings.weight requires_grad= True
bert.embeddings.token_type_embeddings.weight requires_grad= True
bert.embeddings.LayerNorm.weight requires_grad= True
bert.embeddings.LayerNorm.bias requires_grad= True
bert.encoder.layer.0.attention.self.query.weight requires_grad= True
bert.encoder.layer.0.attention.self.query.bias requires_grad= True
bert.encoder.layer.0.attention.self.key.weight requires_grad= True
bert.encoder.layer.0.attention.self.key.bias requires_grad= True
bert.encoder.layer.0.attention.self.value.weight requires_grad= True
bert.encoder.layer.0.attention.self.value.bias requires_grad= True
bert.encoder.layer.0.attention.output.dense.weight requires_grad= True
bert.encoder.layer.0.attention.output.dense.bias requires_grad= True
bert.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.0.intermediate.dense.weight requires_grad= True
bert.encoder.layer.0.intermediate.dense.bias requires_grad= True
bert.encoder.layer.0.output.dense.weight requires_grad= True
bert.encoder.layer.0.output.dense.bias requires_grad= True
bert.encoder.layer.0.output.LayerNorm.weight requires_grad= True
```

```
bert.encoder.layer.0.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.1.attention.self.query.weight requires_grad= True
bert.encoder.layer.1.attention.self.query.bias requires_grad= True
bert.encoder.layer.1.attention.self.key.weight requires_grad= True
bert.encoder.layer.1.attention.self.key.bias requires_grad= True
bert.encoder.layer.1.attention.self.value.weight requires_grad= True
bert.encoder.layer.1.attention.self.value.bias requires_grad= True
bert.encoder.layer.1.attention.output.dense.weight requires_grad= True
bert.encoder.layer.1.attention.output.dense.bias requires_grad= True
bert.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.1.intermediate.dense.weight requires_grad= True
bert.encoder.layer.1.intermediate.dense.bias requires_grad= True
bert.encoder.layer.1.output.dense.weight requires_grad= True
bert.encoder.layer.1.output.dense.bias requires_grad= True
bert.encoder.layer.1.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.1.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.2.attention.self.query.weight requires_grad= True
bert.encoder.layer.2.attention.self.query.bias requires_grad= True
bert.encoder.layer.2.attention.self.key.weight requires_grad= True
bert.encoder.layer.2.attention.self.key.bias requires_grad= True
bert.encoder.layer.2.attention.self.value.weight requires_grad= True
bert.encoder.layer.2.attention.self.value.bias requires_grad= True
bert.encoder.layer.2.attention.output.dense.weight requires_grad= True
bert.encoder.layer.2.attention.output.dense.bias requires_grad= True
bert.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.2.intermediate.dense.weight requires_grad= True
bert.encoder.layer.2.intermediate.dense.bias requires_grad= True
bert.encoder.layer.2.output.dense.weight requires_grad= True
bert.encoder.layer.2.output.dense.bias requires_grad= True
bert.encoder.layer.2.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.2.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.3.attention.self.query.weight requires_grad= True
bert.encoder.layer.3.attention.self.query.bias requires_grad= True
bert.encoder.layer.3.attention.self.key.weight requires_grad= True
bert.encoder.layer.3.attention.self.key.bias requires_grad= True
bert.encoder.layer.3.attention.self.value.weight requires_grad= True
bert.encoder.layer.3.attention.self.value.bias requires_grad= True
bert.encoder.layer.3.attention.output.dense.weight requires_grad= True
bert.encoder.layer.3.attention.output.dense.bias requires_grad= True
bert.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.3.intermediate.dense.weight requires_grad= True
bert.encoder.layer.3.intermediate.dense.bias requires_grad= True
bert.encoder.layer.3.output.dense.weight requires_grad= True
bert.encoder.layer.3.output.dense.bias requires_grad= True
bert.encoder.layer.3.output.LayerNorm.weight requires_grad= True
```

```
bert.encoder.layer.3.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.4.attention.self.query.weight requires_grad= True
bert.encoder.layer.4.attention.self.query.bias requires_grad= True
bert.encoder.layer.4.attention.self.key.weight requires_grad= True
bert.encoder.layer.4.attention.self.key.bias requires_grad= True
bert.encoder.layer.4.attention.self.value.weight requires_grad= True
bert.encoder.layer.4.attention.self.value.bias requires_grad= True
bert.encoder.layer.4.attention.output.dense.weight requires_grad= True
bert.encoder.layer.4.attention.output.dense.bias requires_grad= True
bert.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.4.intermediate.dense.weight requires_grad= True
bert.encoder.layer.4.intermediate.dense.bias requires_grad= True
bert.encoder.layer.4.output.dense.weight requires_grad= True
bert.encoder.layer.4.output.dense.bias requires_grad= True
bert.encoder.layer.4.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.4.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.5.attention.self.query.weight requires_grad= True
bert.encoder.layer.5.attention.self.query.bias requires_grad= True
bert.encoder.layer.5.attention.self.key.weight requires_grad= True
bert.encoder.layer.5.attention.self.key.bias requires_grad= True
bert.encoder.layer.5.attention.self.value.weight requires_grad= True
bert.encoder.layer.5.attention.self.value.bias requires_grad= True
bert.encoder.layer.5.attention.output.dense.weight requires_grad= True
bert.encoder.layer.5.attention.output.dense.bias requires_grad= True
bert.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.5.intermediate.dense.weight requires_grad= True
bert.encoder.layer.5.intermediate.dense.bias requires_grad= True
bert.encoder.layer.5.output.dense.weight requires_grad= True
bert.encoder.layer.5.output.dense.bias requires_grad= True
bert.encoder.layer.5.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.5.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.6.attention.self.query.weight requires_grad= True
bert.encoder.layer.6.attention.self.query.bias requires_grad= True
bert.encoder.layer.6.attention.self.key.weight requires_grad= True
bert.encoder.layer.6.attention.self.key.bias requires_grad= True
bert.encoder.layer.6.attention.self.value.weight requires_grad= True
bert.encoder.layer.6.attention.self.value.bias requires_grad= True
bert.encoder.layer.6.attention.output.dense.weight requires_grad= True
bert.encoder.layer.6.attention.output.dense.bias requires_grad= True
bert.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.6.intermediate.dense.weight requires_grad= True
bert.encoder.layer.6.intermediate.dense.bias requires_grad= True
bert.encoder.layer.6.output.dense.weight requires_grad= True
bert.encoder.layer.6.output.dense.bias requires_grad= True
bert.encoder.layer.6.output.LayerNorm.weight requires_grad= True
```

```
bert.encoder.layer.6.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.7.attention.self.query.weight requires_grad= True
bert.encoder.layer.7.attention.self.query.bias requires_grad= True
bert.encoder.layer.7.attention.self.key.weight requires_grad= True
bert.encoder.layer.7.attention.self.key.bias requires_grad= True
bert.encoder.layer.7.attention.self.value.weight requires_grad= True
bert.encoder.layer.7.attention.self.value.bias requires_grad= True
bert.encoder.layer.7.attention.output.dense.weight requires_grad= True
bert.encoder.layer.7.attention.output.dense.bias requires_grad= True
bert.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.7.intermediate.dense.weight requires_grad= True
bert.encoder.layer.7.intermediate.dense.bias requires_grad= True
bert.encoder.layer.7.output.dense.weight requires_grad= True
bert.encoder.layer.7.output.dense.bias requires_grad= True
bert.encoder.layer.7.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.7.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.8.attention.self.query.weight requires_grad= True
bert.encoder.layer.8.attention.self.query.bias requires_grad= True
bert.encoder.layer.8.attention.self.key.weight requires_grad= True
bert.encoder.layer.8.attention.self.key.bias requires_grad= True
bert.encoder.layer.8.attention.self.value.weight requires_grad= True
bert.encoder.layer.8.attention.self.value.bias requires_grad= True
bert.encoder.layer.8.attention.output.dense.weight requires_grad= True
bert.encoder.layer.8.attention.output.dense.bias requires_grad= True
bert.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.8.intermediate.dense.weight requires_grad= True
bert.encoder.layer.8.intermediate.dense.bias requires_grad= True
bert.encoder.layer.8.output.dense.weight requires_grad= True
bert.encoder.layer.8.output.dense.bias requires_grad= True
bert.encoder.layer.8.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.8.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.9.attention.self.query.weight requires_grad= True
bert.encoder.layer.9.attention.self.query.bias requires_grad= True
bert.encoder.layer.9.attention.self.key.weight requires_grad= True
bert.encoder.layer.9.attention.self.key.bias requires_grad= True
bert.encoder.layer.9.attention.self.value.weight requires_grad= True
bert.encoder.layer.9.attention.self.value.bias requires_grad= True
bert.encoder.layer.9.attention.output.dense.weight requires_grad= True
bert.encoder.layer.9.attention.output.dense.bias requires_grad= True
bert.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.9.intermediate.dense.weight requires_grad= True
bert.encoder.layer.9.intermediate.dense.bias requires_grad= True
bert.encoder.layer.9.output.dense.weight requires_grad= True
bert.encoder.layer.9.output.dense.bias requires_grad= True
bert.encoder.layer.9.output.LayerNorm.weight requires_grad= True
```

```
bert.encoder.layer.9.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.10.attention.self.query.weight requires_grad= True
bert.encoder.layer.10.attention.self.query.bias requires_grad= True
bert.encoder.layer.10.attention.self.key.weight requires_grad= True
bert.encoder.layer.10.attention.self.key.bias requires_grad= True
bert.encoder.layer.10.attention.self.value.weight requires_grad= True
bert.encoder.layer.10.attention.self.value.bias requires_grad= True
bert.encoder.layer.10.attention.output.dense.weight requires_grad= True
bert.encoder.layer.10.attention.output.dense.bias requires_grad= True
bert.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.10.intermediate.dense.weight requires_grad= True
bert.encoder.layer.10.intermediate.dense.bias requires_grad= True
bert.encoder.layer.10.output.dense.weight requires_grad= True
bert.encoder.layer.10.output.dense.bias requires_grad= True
bert.encoder.layer.10.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.10.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.11.attention.self.query.weight requires_grad= True
bert.encoder.layer.11.attention.self.query.bias requires_grad= True
bert.encoder.layer.11.attention.self.key.weight requires_grad= True
bert.encoder.layer.11.attention.self.key.bias requires_grad= True
bert.encoder.layer.11.attention.self.value.weight requires_grad= True
bert.encoder.layer.11.attention.self.value.bias requires_grad= True
bert.encoder.layer.11.attention.output.dense.weight requires_grad= True
bert.encoder.layer.11.attention.output.dense.bias requires_grad= True
bert.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.11.intermediate.dense.weight requires_grad= True
bert.encoder.layer.11.intermediate.dense.bias requires_grad= True
bert.encoder.layer.11.output.dense.weight requires_grad= True
bert.encoder.layer.11.output.dense.bias requires_grad= True
bert.encoder.layer.11.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.11.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.12.attention.self.query.weight requires_grad= True
bert.encoder.layer.12.attention.self.query.bias requires_grad= True
bert.encoder.layer.12.attention.self.key.weight requires_grad= True
bert.encoder.layer.12.attention.self.key.bias requires_grad= True
bert.encoder.layer.12.attention.self.value.weight requires_grad= True
bert.encoder.layer.12.attention.self.value.bias requires_grad= True
bert.encoder.layer.12.attention.output.dense.weight requires_grad= True
bert.encoder.layer.12.attention.output.dense.bias requires_grad= True
bert.encoder.layer.12.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.12.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.12.intermediate.dense.weight requires_grad= True
bert.encoder.layer.12.intermediate.dense.bias requires_grad= True
bert.encoder.layer.12.output.dense.weight requires_grad= True
bert.encoder.layer.12.output.dense.bias requires_grad= True
bert.encoder.layer.12.output.LayerNorm.weight requires_grad= True
```

```
bert.encoder.layer.12.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.13.attention.self.query.weight requires_grad= True
bert.encoder.layer.13.attention.self.query.bias requires_grad= True
bert.encoder.layer.13.attention.self.key.weight requires_grad= True
bert.encoder.layer.13.attention.self.key.bias requires_grad= True
bert.encoder.layer.13.attention.self.value.weight requires_grad= True
bert.encoder.layer.13.attention.self.value.bias requires_grad= True
bert.encoder.layer.13.attention.output.dense.weight requires_grad= True
bert.encoder.layer.13.attention.output.dense.bias requires_grad= True
bert.encoder.layer.13.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.13.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.13.intermediate.dense.weight requires_grad= True
bert.encoder.layer.13.intermediate.dense.bias requires_grad= True
bert.encoder.layer.13.output.dense.weight requires_grad= True
bert.encoder.layer.13.output.dense.bias requires_grad= True
bert.encoder.layer.13.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.13.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.14.attention.self.query.weight requires_grad= True
bert.encoder.layer.14.attention.self.query.bias requires_grad= True
bert.encoder.layer.14.attention.self.key.weight requires_grad= True
bert.encoder.layer.14.attention.self.key.bias requires_grad= True
bert.encoder.layer.14.attention.self.value.weight requires_grad= True
bert.encoder.layer.14.attention.self.value.bias requires_grad= True
bert.encoder.layer.14.attention.output.dense.weight requires_grad= True
bert.encoder.layer.14.attention.output.dense.bias requires_grad= True
bert.encoder.layer.14.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.14.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.14.intermediate.dense.weight requires_grad= True
bert.encoder.layer.14.intermediate.dense.bias requires_grad= True
bert.encoder.layer.14.output.dense.weight requires_grad= True
bert.encoder.layer.14.output.dense.bias requires_grad= True
bert.encoder.layer.14.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.14.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.15.attention.self.query.weight requires_grad= True
bert.encoder.layer.15.attention.self.query.bias requires_grad= True
bert.encoder.layer.15.attention.self.key.weight requires_grad= True
bert.encoder.layer.15.attention.self.key.bias requires_grad= True
bert.encoder.layer.15.attention.self.value.weight requires_grad= True
bert.encoder.layer.15.attention.self.value.bias requires_grad= True
bert.encoder.layer.15.attention.output.dense.weight requires_grad= True
bert.encoder.layer.15.attention.output.dense.bias requires_grad= True
bert.encoder.layer.15.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.15.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.15.intermediate.dense.weight requires_grad= True
bert.encoder.layer.15.intermediate.dense.bias requires_grad= True
bert.encoder.layer.15.output.dense.weight requires_grad= True
bert.encoder.layer.15.output.dense.bias requires_grad= True
bert.encoder.layer.15.output.LayerNorm.weight requires_grad= True
```

```
bert.encoder.layer.15.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.16.attention.self.query.weight requires_grad= True
bert.encoder.layer.16.attention.self.query.bias requires_grad= True
bert.encoder.layer.16.attention.self.key.weight requires_grad= True
bert.encoder.layer.16.attention.self.key.bias requires_grad= True
bert.encoder.layer.16.attention.self.value.weight requires_grad= True
bert.encoder.layer.16.attention.self.value.bias requires_grad= True
bert.encoder.layer.16.attention.output.dense.weight requires_grad= True
bert.encoder.layer.16.attention.output.dense.bias requires_grad= True
bert.encoder.layer.16.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.16.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.16.intermediate.dense.weight requires_grad= True
bert.encoder.layer.16.intermediate.dense.bias requires_grad= True
bert.encoder.layer.16.output.dense.weight requires_grad= True
bert.encoder.layer.16.output.dense.bias requires_grad= True
bert.encoder.layer.16.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.16.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.17.attention.self.query.weight requires_grad= True
bert.encoder.layer.17.attention.self.query.bias requires_grad= True
bert.encoder.layer.17.attention.self.key.weight requires_grad= True
bert.encoder.layer.17.attention.self.key.bias requires_grad= True
bert.encoder.layer.17.attention.self.value.weight requires_grad= True
bert.encoder.layer.17.attention.self.value.bias requires_grad= True
bert.encoder.layer.17.attention.output.dense.weight requires_grad= True
bert.encoder.layer.17.attention.output.dense.bias requires_grad= True
bert.encoder.layer.17.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.17.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.17.intermediate.dense.weight requires_grad= True
bert.encoder.layer.17.intermediate.dense.bias requires_grad= True
bert.encoder.layer.17.output.dense.weight requires_grad= True
bert.encoder.layer.17.output.dense.bias requires_grad= True
bert.encoder.layer.17.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.17.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.18.attention.self.query.weight requires_grad= True
bert.encoder.layer.18.attention.self.query.bias requires_grad= True
bert.encoder.layer.18.attention.self.key.weight requires_grad= True
bert.encoder.layer.18.attention.self.key.bias requires_grad= True
bert.encoder.layer.18.attention.self.value.weight requires_grad= True
bert.encoder.layer.18.attention.self.value.bias requires_grad= True
bert.encoder.layer.18.attention.output.dense.weight requires_grad= True
bert.encoder.layer.18.attention.output.dense.bias requires_grad= True
bert.encoder.layer.18.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.18.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.18.intermediate.dense.weight requires_grad= True
bert.encoder.layer.18.intermediate.dense.bias requires_grad= True
bert.encoder.layer.18.output.dense.weight requires_grad= True
bert.encoder.layer.18.output.dense.bias requires_grad= True
bert.encoder.layer.18.output.LayerNorm.weight requires_grad= True
```

```
bert.encoder.layer.18.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.19.attention.self.query.weight requires_grad= True
bert.encoder.layer.19.attention.self.query.bias requires_grad= True
bert.encoder.layer.19.attention.self.key.weight requires_grad= True
bert.encoder.layer.19.attention.self.key.bias requires_grad= True
bert.encoder.layer.19.attention.self.value.weight requires_grad= True
bert.encoder.layer.19.attention.self.value.bias requires_grad= True
bert.encoder.layer.19.attention.output.dense.weight requires_grad= True
bert.encoder.layer.19.attention.output.dense.bias requires_grad= True
bert.encoder.layer.19.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.19.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.19.intermediate.dense.weight requires_grad= True
bert.encoder.layer.19.intermediate.dense.bias requires_grad= True
bert.encoder.layer.19.output.dense.weight requires_grad= True
bert.encoder.layer.19.output.dense.bias requires_grad= True
bert.encoder.layer.19.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.19.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.20.attention.self.query.weight requires_grad= True
bert.encoder.layer.20.attention.self.query.bias requires_grad= True
bert.encoder.layer.20.attention.self.key.weight requires_grad= True
bert.encoder.layer.20.attention.self.key.bias requires_grad= True
bert.encoder.layer.20.attention.self.value.weight requires_grad= True
bert.encoder.layer.20.attention.self.value.bias requires_grad= True
bert.encoder.layer.20.attention.output.dense.weight requires_grad= True
bert.encoder.layer.20.attention.output.dense.bias requires_grad= True
bert.encoder.layer.20.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.20.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.20.intermediate.dense.weight requires_grad= True
bert.encoder.layer.20.intermediate.dense.bias requires_grad= True
bert.encoder.layer.20.output.dense.weight requires_grad= True
bert.encoder.layer.20.output.dense.bias requires_grad= True
bert.encoder.layer.20.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.20.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.21.attention.self.query.weight requires_grad= True
bert.encoder.layer.21.attention.self.query.bias requires_grad= True
bert.encoder.layer.21.attention.self.key.weight requires_grad= True
bert.encoder.layer.21.attention.self.key.bias requires_grad= True
bert.encoder.layer.21.attention.self.value.weight requires_grad= True
bert.encoder.layer.21.attention.self.value.bias requires_grad= True
bert.encoder.layer.21.attention.output.dense.weight requires_grad= True
bert.encoder.layer.21.attention.output.dense.bias requires_grad= True
bert.encoder.layer.21.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.21.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.21.intermediate.dense.weight requires_grad= True
bert.encoder.layer.21.intermediate.dense.bias requires_grad= True
bert.encoder.layer.21.output.dense.weight requires_grad= True
bert.encoder.layer.21.output.dense.bias requires_grad= True
bert.encoder.layer.21.output.LayerNorm.weight requires_grad= True
```

```
bert.encoder.layer.21.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.22.attention.self.query.weight requires_grad= True
bert.encoder.layer.22.attention.self.query.bias requires_grad= True
bert.encoder.layer.22.attention.self.key.weight requires_grad= True
bert.encoder.layer.22.attention.self.key.bias requires_grad= True
bert.encoder.layer.22.attention.self.value.weight requires_grad= True
bert.encoder.layer.22.attention.self.value.bias requires_grad= True
bert.encoder.layer.22.attention.output.dense.weight requires_grad= True
bert.encoder.layer.22.attention.output.dense.bias requires_grad= True
bert.encoder.layer.22.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.22.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.22.intermediate.dense.weight requires_grad= True
bert.encoder.layer.22.intermediate.dense.bias requires_grad= True
bert.encoder.layer.22.output.dense.weight requires_grad= True
bert.encoder.layer.22.output.dense.bias requires_grad= True
bert.encoder.layer.22.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.22.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.23.attention.self.query.weight requires_grad= True
bert.encoder.layer.23.attention.self.query.bias requires_grad= True
bert.encoder.layer.23.attention.self.key.weight requires_grad= True
bert.encoder.layer.23.attention.self.key.bias requires_grad= True
bert.encoder.layer.23.attention.self.value.weight requires_grad= True
bert.encoder.layer.23.attention.self.value.bias requires_grad= True
bert.encoder.layer.23.attention.output.dense.weight requires_grad= True
bert.encoder.layer.23.attention.output.dense.bias requires_grad= True
bert.encoder.layer.23.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.23.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.23.intermediate.dense.weight requires_grad= True
bert.encoder.layer.23.intermediate.dense.bias requires_grad= True
bert.encoder.layer.23.output.dense.weight requires_grad= True
bert.encoder.layer.23.output.dense.bias requires_grad= True
bert.encoder.layer.23.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.23.output.LayerNorm.bias requires_grad= True
bert.pooler.dense.weight requires_grad= True
bert.pooler.dense.bias requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True
```

[ ]:
```python
########################################
layers_to_unfreeze = [
    "bert.encoder.layer.23.",
    "bert.pooler.",
    "classifier.",
]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
```

```
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("==============")
########################################
print("Experiment configuration used with this experiment:")
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
########################################
print("==============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```


```
BertConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "type_vocab_size": 2,
  "use_cache": true,
```

```
  "vocab_size": 28996
}

=============
num_parameters: 333581314
num_trainable_parameters: 13647874
=============
Experiment configuration used with this experiment:
model used: bert-large-cased
learning rate used: 5e-06
number of epochs: 1
maximum sequence length: 128
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
=============
num_trainable_parameters: 13647874
```

[ ]:
```
model.resize_token_embeddings(len(tokenizer))
```

[ ]:
```
Embedding(28996, 1024, padding_idx=0)
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
bert.embeddings.word_embeddings.weight requires_grad= False
bert.embeddings.position_embeddings.weight requires_grad= False
bert.embeddings.token_type_embeddings.weight requires_grad= False
bert.embeddings.LayerNorm.weight requires_grad= False
bert.embeddings.LayerNorm.bias requires_grad= False
bert.encoder.layer.0.attention.self.query.weight requires_grad= False
bert.encoder.layer.0.attention.self.query.bias requires_grad= False
bert.encoder.layer.0.attention.self.key.weight requires_grad= False
bert.encoder.layer.0.attention.self.key.bias requires_grad= False
bert.encoder.layer.0.attention.self.value.weight requires_grad= False
bert.encoder.layer.0.attention.self.value.bias requires_grad= False
bert.encoder.layer.0.attention.output.dense.weight requires_grad= False
bert.encoder.layer.0.attention.output.dense.bias requires_grad= False
bert.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.0.intermediate.dense.weight requires_grad= False
bert.encoder.layer.0.intermediate.dense.bias requires_grad= False
bert.encoder.layer.0.output.dense.weight requires_grad= False
bert.encoder.layer.0.output.dense.bias requires_grad= False
bert.encoder.layer.0.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.0.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.1.attention.self.query.weight requires_grad= False
bert.encoder.layer.1.attention.self.query.bias requires_grad= False
bert.encoder.layer.1.attention.self.key.weight requires_grad= False
bert.encoder.layer.1.attention.self.key.bias requires_grad= False
bert.encoder.layer.1.attention.self.value.weight requires_grad= False
bert.encoder.layer.1.attention.self.value.bias requires_grad= False
bert.encoder.layer.1.attention.output.dense.weight requires_grad= False
bert.encoder.layer.1.attention.output.dense.bias requires_grad= False
bert.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.1.intermediate.dense.weight requires_grad= False
bert.encoder.layer.1.intermediate.dense.bias requires_grad= False
bert.encoder.layer.1.output.dense.weight requires_grad= False
bert.encoder.layer.1.output.dense.bias requires_grad= False
bert.encoder.layer.1.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.1.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.2.attention.self.query.weight requires_grad= False
bert.encoder.layer.2.attention.self.query.bias requires_grad= False
bert.encoder.layer.2.attention.self.key.weight requires_grad= False
bert.encoder.layer.2.attention.self.key.bias requires_grad= False
bert.encoder.layer.2.attention.self.value.weight requires_grad= False
bert.encoder.layer.2.attention.self.value.bias requires_grad= False
bert.encoder.layer.2.attention.output.dense.weight requires_grad= False
bert.encoder.layer.2.attention.output.dense.bias requires_grad= False
bert.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.2.intermediate.dense.weight requires_grad= False
bert.encoder.layer.2.intermediate.dense.bias requires_grad= False
bert.encoder.layer.2.output.dense.weight requires_grad= False
bert.encoder.layer.2.output.dense.bias requires_grad= False
bert.encoder.layer.2.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.2.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.3.attention.self.query.weight requires_grad= False
bert.encoder.layer.3.attention.self.query.bias requires_grad= False
bert.encoder.layer.3.attention.self.key.weight requires_grad= False
bert.encoder.layer.3.attention.self.key.bias requires_grad= False
bert.encoder.layer.3.attention.self.value.weight requires_grad= False
bert.encoder.layer.3.attention.self.value.bias requires_grad= False
bert.encoder.layer.3.attention.output.dense.weight requires_grad= False
bert.encoder.layer.3.attention.output.dense.bias requires_grad= False
bert.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.3.intermediate.dense.weight requires_grad= False
bert.encoder.layer.3.intermediate.dense.bias requires_grad= False
bert.encoder.layer.3.output.dense.weight requires_grad= False
bert.encoder.layer.3.output.dense.bias requires_grad= False
bert.encoder.layer.3.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.3.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.4.attention.self.query.weight requires_grad= False
bert.encoder.layer.4.attention.self.query.bias requires_grad= False
bert.encoder.layer.4.attention.self.key.weight requires_grad= False
bert.encoder.layer.4.attention.self.key.bias requires_grad= False
bert.encoder.layer.4.attention.self.value.weight requires_grad= False
bert.encoder.layer.4.attention.self.value.bias requires_grad= False
bert.encoder.layer.4.attention.output.dense.weight requires_grad= False
bert.encoder.layer.4.attention.output.dense.bias requires_grad= False
bert.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.4.intermediate.dense.weight requires_grad= False
bert.encoder.layer.4.intermediate.dense.bias requires_grad= False
bert.encoder.layer.4.output.dense.weight requires_grad= False
bert.encoder.layer.4.output.dense.bias requires_grad= False
bert.encoder.layer.4.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.4.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.5.attention.self.query.weight requires_grad= False
bert.encoder.layer.5.attention.self.query.bias requires_grad= False
bert.encoder.layer.5.attention.self.key.weight requires_grad= False
bert.encoder.layer.5.attention.self.key.bias requires_grad= False
bert.encoder.layer.5.attention.self.value.weight requires_grad= False
bert.encoder.layer.5.attention.self.value.bias requires_grad= False
bert.encoder.layer.5.attention.output.dense.weight requires_grad= False
bert.encoder.layer.5.attention.output.dense.bias requires_grad= False
bert.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.5.intermediate.dense.weight requires_grad= False
bert.encoder.layer.5.intermediate.dense.bias requires_grad= False
bert.encoder.layer.5.output.dense.weight requires_grad= False
bert.encoder.layer.5.output.dense.bias requires_grad= False
bert.encoder.layer.5.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.5.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.6.attention.self.query.weight requires_grad= False
bert.encoder.layer.6.attention.self.query.bias requires_grad= False
bert.encoder.layer.6.attention.self.key.weight requires_grad= False
bert.encoder.layer.6.attention.self.key.bias requires_grad= False
bert.encoder.layer.6.attention.self.value.weight requires_grad= False
bert.encoder.layer.6.attention.self.value.bias requires_grad= False
bert.encoder.layer.6.attention.output.dense.weight requires_grad= False
bert.encoder.layer.6.attention.output.dense.bias requires_grad= False
bert.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.6.intermediate.dense.weight requires_grad= False
bert.encoder.layer.6.intermediate.dense.bias requires_grad= False
bert.encoder.layer.6.output.dense.weight requires_grad= False
bert.encoder.layer.6.output.dense.bias requires_grad= False
bert.encoder.layer.6.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.6.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.7.attention.self.query.weight requires_grad= False
bert.encoder.layer.7.attention.self.query.bias requires_grad= False
bert.encoder.layer.7.attention.self.key.weight requires_grad= False
bert.encoder.layer.7.attention.self.key.bias requires_grad= False
bert.encoder.layer.7.attention.self.value.weight requires_grad= False
bert.encoder.layer.7.attention.self.value.bias requires_grad= False
bert.encoder.layer.7.attention.output.dense.weight requires_grad= False
bert.encoder.layer.7.attention.output.dense.bias requires_grad= False
bert.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.7.intermediate.dense.weight requires_grad= False
bert.encoder.layer.7.intermediate.dense.bias requires_grad= False
bert.encoder.layer.7.output.dense.weight requires_grad= False
bert.encoder.layer.7.output.dense.bias requires_grad= False
bert.encoder.layer.7.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.7.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.8.attention.self.query.weight requires_grad= False
bert.encoder.layer.8.attention.self.query.bias requires_grad= False
bert.encoder.layer.8.attention.self.key.weight requires_grad= False
bert.encoder.layer.8.attention.self.key.bias requires_grad= False
bert.encoder.layer.8.attention.self.value.weight requires_grad= False
bert.encoder.layer.8.attention.self.value.bias requires_grad= False
bert.encoder.layer.8.attention.output.dense.weight requires_grad= False
bert.encoder.layer.8.attention.output.dense.bias requires_grad= False
bert.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.8.intermediate.dense.weight requires_grad= False
bert.encoder.layer.8.intermediate.dense.bias requires_grad= False
bert.encoder.layer.8.output.dense.weight requires_grad= False
bert.encoder.layer.8.output.dense.bias requires_grad= False
bert.encoder.layer.8.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.8.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.9.attention.self.query.weight requires_grad= False
bert.encoder.layer.9.attention.self.query.bias requires_grad= False
bert.encoder.layer.9.attention.self.key.weight requires_grad= False
bert.encoder.layer.9.attention.self.key.bias requires_grad= False
bert.encoder.layer.9.attention.self.value.weight requires_grad= False
bert.encoder.layer.9.attention.self.value.bias requires_grad= False
bert.encoder.layer.9.attention.output.dense.weight requires_grad= False
bert.encoder.layer.9.attention.output.dense.bias requires_grad= False
bert.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.9.intermediate.dense.weight requires_grad= False
bert.encoder.layer.9.intermediate.dense.bias requires_grad= False
bert.encoder.layer.9.output.dense.weight requires_grad= False
bert.encoder.layer.9.output.dense.bias requires_grad= False
bert.encoder.layer.9.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.9.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.10.attention.self.query.weight requires_grad= False
bert.encoder.layer.10.attention.self.query.bias requires_grad= False
bert.encoder.layer.10.attention.self.key.weight requires_grad= False
bert.encoder.layer.10.attention.self.key.bias requires_grad= False
bert.encoder.layer.10.attention.self.value.weight requires_grad= False
bert.encoder.layer.10.attention.self.value.bias requires_grad= False
bert.encoder.layer.10.attention.output.dense.weight requires_grad= False
bert.encoder.layer.10.attention.output.dense.bias requires_grad= False
bert.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.10.intermediate.dense.weight requires_grad= False
bert.encoder.layer.10.intermediate.dense.bias requires_grad= False
bert.encoder.layer.10.output.dense.weight requires_grad= False
bert.encoder.layer.10.output.dense.bias requires_grad= False
bert.encoder.layer.10.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.10.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.11.attention.self.query.weight requires_grad= False
bert.encoder.layer.11.attention.self.query.bias requires_grad= False
bert.encoder.layer.11.attention.self.key.weight requires_grad= False
bert.encoder.layer.11.attention.self.key.bias requires_grad= False
bert.encoder.layer.11.attention.self.value.weight requires_grad= False
bert.encoder.layer.11.attention.self.value.bias requires_grad= False
bert.encoder.layer.11.attention.output.dense.weight requires_grad= False
bert.encoder.layer.11.attention.output.dense.bias requires_grad= False
bert.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.11.intermediate.dense.weight requires_grad= False
bert.encoder.layer.11.intermediate.dense.bias requires_grad= False
bert.encoder.layer.11.output.dense.weight requires_grad= False
bert.encoder.layer.11.output.dense.bias requires_grad= False
bert.encoder.layer.11.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.11.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.12.attention.self.query.weight requires_grad= False
bert.encoder.layer.12.attention.self.query.bias requires_grad= False
bert.encoder.layer.12.attention.self.key.weight requires_grad= False
bert.encoder.layer.12.attention.self.key.bias requires_grad= False
bert.encoder.layer.12.attention.self.value.weight requires_grad= False
bert.encoder.layer.12.attention.self.value.bias requires_grad= False
bert.encoder.layer.12.attention.output.dense.weight requires_grad= False
bert.encoder.layer.12.attention.output.dense.bias requires_grad= False
bert.encoder.layer.12.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.12.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.12.intermediate.dense.weight requires_grad= False
bert.encoder.layer.12.intermediate.dense.bias requires_grad= False
bert.encoder.layer.12.output.dense.weight requires_grad= False
bert.encoder.layer.12.output.dense.bias requires_grad= False
bert.encoder.layer.12.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.12.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.13.attention.self.query.weight requires_grad= False
bert.encoder.layer.13.attention.self.query.bias requires_grad= False
bert.encoder.layer.13.attention.self.key.weight requires_grad= False
bert.encoder.layer.13.attention.self.key.bias requires_grad= False
bert.encoder.layer.13.attention.self.value.weight requires_grad= False
bert.encoder.layer.13.attention.self.value.bias requires_grad= False
bert.encoder.layer.13.attention.output.dense.weight requires_grad= False
bert.encoder.layer.13.attention.output.dense.bias requires_grad= False
bert.encoder.layer.13.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.13.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.13.intermediate.dense.weight requires_grad= False
bert.encoder.layer.13.intermediate.dense.bias requires_grad= False
bert.encoder.layer.13.output.dense.weight requires_grad= False
bert.encoder.layer.13.output.dense.bias requires_grad= False
bert.encoder.layer.13.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.13.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.14.attention.self.query.weight requires_grad= False
bert.encoder.layer.14.attention.self.query.bias requires_grad= False
bert.encoder.layer.14.attention.self.key.weight requires_grad= False
bert.encoder.layer.14.attention.self.key.bias requires_grad= False
bert.encoder.layer.14.attention.self.value.weight requires_grad= False
bert.encoder.layer.14.attention.self.value.bias requires_grad= False
bert.encoder.layer.14.attention.output.dense.weight requires_grad= False
bert.encoder.layer.14.attention.output.dense.bias requires_grad= False
bert.encoder.layer.14.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.14.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.14.intermediate.dense.weight requires_grad= False
bert.encoder.layer.14.intermediate.dense.bias requires_grad= False
bert.encoder.layer.14.output.dense.weight requires_grad= False
bert.encoder.layer.14.output.dense.bias requires_grad= False
bert.encoder.layer.14.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.14.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.15.attention.self.query.weight requires_grad= False
bert.encoder.layer.15.attention.self.query.bias requires_grad= False
bert.encoder.layer.15.attention.self.key.weight requires_grad= False
bert.encoder.layer.15.attention.self.key.bias requires_grad= False
bert.encoder.layer.15.attention.self.value.weight requires_grad= False
bert.encoder.layer.15.attention.self.value.bias requires_grad= False
bert.encoder.layer.15.attention.output.dense.weight requires_grad= False
bert.encoder.layer.15.attention.output.dense.bias requires_grad= False
bert.encoder.layer.15.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.15.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.15.intermediate.dense.weight requires_grad= False
bert.encoder.layer.15.intermediate.dense.bias requires_grad= False
bert.encoder.layer.15.output.dense.weight requires_grad= False
bert.encoder.layer.15.output.dense.bias requires_grad= False
bert.encoder.layer.15.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.15.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.16.attention.self.query.weight requires_grad= False
bert.encoder.layer.16.attention.self.query.bias requires_grad= False
bert.encoder.layer.16.attention.self.key.weight requires_grad= False
bert.encoder.layer.16.attention.self.key.bias requires_grad= False
bert.encoder.layer.16.attention.self.value.weight requires_grad= False
bert.encoder.layer.16.attention.self.value.bias requires_grad= False
bert.encoder.layer.16.attention.output.dense.weight requires_grad= False
bert.encoder.layer.16.attention.output.dense.bias requires_grad= False
bert.encoder.layer.16.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.16.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.16.intermediate.dense.weight requires_grad= False
bert.encoder.layer.16.intermediate.dense.bias requires_grad= False
bert.encoder.layer.16.output.dense.weight requires_grad= False
bert.encoder.layer.16.output.dense.bias requires_grad= False
bert.encoder.layer.16.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.16.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.17.attention.self.query.weight requires_grad= False
bert.encoder.layer.17.attention.self.query.bias requires_grad= False
bert.encoder.layer.17.attention.self.key.weight requires_grad= False
bert.encoder.layer.17.attention.self.key.bias requires_grad= False
bert.encoder.layer.17.attention.self.value.weight requires_grad= False
bert.encoder.layer.17.attention.self.value.bias requires_grad= False
bert.encoder.layer.17.attention.output.dense.weight requires_grad= False
bert.encoder.layer.17.attention.output.dense.bias requires_grad= False
bert.encoder.layer.17.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.17.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.17.intermediate.dense.weight requires_grad= False
bert.encoder.layer.17.intermediate.dense.bias requires_grad= False
bert.encoder.layer.17.output.dense.weight requires_grad= False
bert.encoder.layer.17.output.dense.bias requires_grad= False
bert.encoder.layer.17.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.17.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.18.attention.self.query.weight requires_grad= False
bert.encoder.layer.18.attention.self.query.bias requires_grad= False
bert.encoder.layer.18.attention.self.key.weight requires_grad= False
bert.encoder.layer.18.attention.self.key.bias requires_grad= False
bert.encoder.layer.18.attention.self.value.weight requires_grad= False
bert.encoder.layer.18.attention.self.value.bias requires_grad= False
bert.encoder.layer.18.attention.output.dense.weight requires_grad= False
bert.encoder.layer.18.attention.output.dense.bias requires_grad= False
bert.encoder.layer.18.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.18.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.18.intermediate.dense.weight requires_grad= False
bert.encoder.layer.18.intermediate.dense.bias requires_grad= False
bert.encoder.layer.18.output.dense.weight requires_grad= False
bert.encoder.layer.18.output.dense.bias requires_grad= False
bert.encoder.layer.18.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.18.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.19.attention.self.query.weight requires_grad= False
bert.encoder.layer.19.attention.self.query.bias requires_grad= False
bert.encoder.layer.19.attention.self.key.weight requires_grad= False
bert.encoder.layer.19.attention.self.key.bias requires_grad= False
bert.encoder.layer.19.attention.self.value.weight requires_grad= False
bert.encoder.layer.19.attention.self.value.bias requires_grad= False
bert.encoder.layer.19.attention.output.dense.weight requires_grad= False
bert.encoder.layer.19.attention.output.dense.bias requires_grad= False
bert.encoder.layer.19.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.19.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.19.intermediate.dense.weight requires_grad= False
bert.encoder.layer.19.intermediate.dense.bias requires_grad= False
bert.encoder.layer.19.output.dense.weight requires_grad= False
bert.encoder.layer.19.output.dense.bias requires_grad= False
bert.encoder.layer.19.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.19.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.20.attention.self.query.weight requires_grad= False
bert.encoder.layer.20.attention.self.query.bias requires_grad= False
bert.encoder.layer.20.attention.self.key.weight requires_grad= False
bert.encoder.layer.20.attention.self.key.bias requires_grad= False
bert.encoder.layer.20.attention.self.value.weight requires_grad= False
bert.encoder.layer.20.attention.self.value.bias requires_grad= False
bert.encoder.layer.20.attention.output.dense.weight requires_grad= False
bert.encoder.layer.20.attention.output.dense.bias requires_grad= False
bert.encoder.layer.20.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.20.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.20.intermediate.dense.weight requires_grad= False
bert.encoder.layer.20.intermediate.dense.bias requires_grad= False
bert.encoder.layer.20.output.dense.weight requires_grad= False
bert.encoder.layer.20.output.dense.bias requires_grad= False
bert.encoder.layer.20.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.20.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.21.attention.self.query.weight requires_grad= False
bert.encoder.layer.21.attention.self.query.bias requires_grad= False
bert.encoder.layer.21.attention.self.key.weight requires_grad= False
bert.encoder.layer.21.attention.self.key.bias requires_grad= False
bert.encoder.layer.21.attention.self.value.weight requires_grad= False
bert.encoder.layer.21.attention.self.value.bias requires_grad= False
bert.encoder.layer.21.attention.output.dense.weight requires_grad= False
bert.encoder.layer.21.attention.output.dense.bias requires_grad= False
bert.encoder.layer.21.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.21.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.21.intermediate.dense.weight requires_grad= False
bert.encoder.layer.21.intermediate.dense.bias requires_grad= False
bert.encoder.layer.21.output.dense.weight requires_grad= False
bert.encoder.layer.21.output.dense.bias requires_grad= False
bert.encoder.layer.21.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.21.output.LayerNorm.bias requires_grad= False
```

```
bert.encoder.layer.22.attention.self.query.weight requires_grad= False
bert.encoder.layer.22.attention.self.query.bias requires_grad= False
bert.encoder.layer.22.attention.self.key.weight requires_grad= False
bert.encoder.layer.22.attention.self.key.bias requires_grad= False
bert.encoder.layer.22.attention.self.value.weight requires_grad= False
bert.encoder.layer.22.attention.self.value.bias requires_grad= False
bert.encoder.layer.22.attention.output.dense.weight requires_grad= False
bert.encoder.layer.22.attention.output.dense.bias requires_grad= False
bert.encoder.layer.22.attention.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.22.attention.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.22.intermediate.dense.weight requires_grad= False
bert.encoder.layer.22.intermediate.dense.bias requires_grad= False
bert.encoder.layer.22.output.dense.weight requires_grad= False
bert.encoder.layer.22.output.dense.bias requires_grad= False
bert.encoder.layer.22.output.LayerNorm.weight requires_grad= False
bert.encoder.layer.22.output.LayerNorm.bias requires_grad= False
bert.encoder.layer.23.attention.self.query.weight requires_grad= True
bert.encoder.layer.23.attention.self.query.bias requires_grad= True
bert.encoder.layer.23.attention.self.key.weight requires_grad= True
bert.encoder.layer.23.attention.self.key.bias requires_grad= True
bert.encoder.layer.23.attention.self.value.weight requires_grad= True
bert.encoder.layer.23.attention.self.value.bias requires_grad= True
bert.encoder.layer.23.attention.output.dense.weight requires_grad= True
bert.encoder.layer.23.attention.output.dense.bias requires_grad= True
bert.encoder.layer.23.attention.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.23.attention.output.LayerNorm.bias requires_grad= True
bert.encoder.layer.23.intermediate.dense.weight requires_grad= True
bert.encoder.layer.23.intermediate.dense.bias requires_grad= True
bert.encoder.layer.23.output.dense.weight requires_grad= True
bert.encoder.layer.23.output.dense.bias requires_grad= True
bert.encoder.layer.23.output.LayerNorm.weight requires_grad= True
bert.encoder.layer.23.output.LayerNorm.bias requires_grad= True
bert.pooler.dense.weight requires_grad= True
bert.pooler.dense.bias requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True
```

[ ]:
```
model.resize_token_embeddings(len(tokenizer))
```

[ ]:
```
Embedding(28996, 1024, padding_idx=0)
```

[ ]:
```
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
    val_dataset = val_data_hf,
    output_dir = dir_results,
```

```
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-20-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 0.6936668157577515, 'eval_accuracy':
0.47268408551068886, 'eval_precision': 0.4318181818181818, 'eval_recall':
0.4947916666666667, 'eval_f1': 0.46116504854368934, 'eval_runtime': 2.982,
'eval_samples_per_second': 141.182, 'eval_steps_per_second': 1.341, 'epoch':
1.0}
Test metrics: {'eval_loss': 0.6971972584724426, 'eval_accuracy':
0.48745910577971646, 'eval_precision': 0.4684095860566449, 'eval_recall':
0.4875283446712018, 'eval_f1': 0.4777777777777778, 'eval_runtime': 5.2386,
'eval_samples_per_second': 175.047, 'eval_steps_per_second': 1.527, 'epoch':
1.0}
```

[ ]:
```
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
  ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
```

```
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to: /content/drive/MyDrive/266-final/models/single_bert-
large-cased_binary_complexity_20250411_121110

<IPython.core.display.HTML object>

EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```

0.2.6 snc roberta-large regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[ ]:
```
# Define Experiment Parameters
# named_model = "bert-base-cased"
# named_model = "roberta-base"
# named_model = "bert-large-cased"
named_model = "roberta-large"
# named_model = "" # modern bert
#############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
num_epochs = 1
##########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
#############
y_col = "binary_complexity"
# y_col = "complexity"
#############
x_task = "single"
```

```
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
########################################
# custom_config = BertConfig.from_pretrained("roberta-base")
# custom_config.hidden_act = "gelu"  # alts: "relu" "silu"
# custom_config.attention_probs_dropout_prob = 0.1
# custom_config.hidden_dropout_prob = 0.1
# custom_config.gradient_checkpointing = False
########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="roberta-large",
    local_model_path=None,
    config=None)
############
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
#     local_model_path="...CONFIGURE_PATH...",
```

```
#     config=custom_config)
print("=============")
print(named_model, ":")
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters at load:", model.
  ↪num_parameters(only_trainable=True))
print("=============")
print("model lineage:", MODEL_LINEAGE)
print("=============")
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]

Map:   0%|          | 0/421 [00:00<?, ? examples/s]

Map:   0%|          | 0/917 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([  101,  1252,  1106,  1103,  3824,
1104, 19892, 11220,  1324,  1119,
         1522,  3839,   117,  1272,  1103,  1555,  1104,  1103, 11563,  5609,
         1106,  1172,   132,  1152,  2446,  1122,  1113,  1147,  3221,   119,
          102,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0]),
'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])}
Loading from Hugging Face model: roberta-large

tokenizer_config.json:   0%|          | 0.00/25.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/482 [00:00<?, ?B/s]

vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]

merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.36M [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/1.42G [00:00<?, ?B/s]
```

```
Some weights of RobertaForSequenceClassification were not initialized from the
model checkpoint at roberta-large and are newly initialized:
['classifier.dense.bias', 'classifier.dense.weight', 'classifier.out_proj.bias',
'classifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

=============
roberta-large :
=============
num_parameters: 355361794
num_trainable_parameters at load: 355361794
=============
model lineage: {'type': 'huggingface_hub', 'path': 'roberta-large', 'timestamp':
'2025-04-11 12:12:07'}
=============
```

[ ]:
```
print(model)
```

```
RobertaForSequenceClassification(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(50265, 1024, padding_idx=1)
      (position_embeddings): Embedding(514, 1024, padding_idx=1)
      (token_type_embeddings): Embedding(1, 1024)
      (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0-23): 24 x RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSdpaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-05,
elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
```

```
          (output): RobertaOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (classifier): RobertaClassificationHead(
    (dense): Linear(in_features=1024, out_features=1024, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
    (out_proj): Linear(in_features=1024, out_features=2, bias=True)
  )
)
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
roberta.embeddings.word_embeddings.weight requires_grad= True
roberta.embeddings.position_embeddings.weight requires_grad= True
roberta.embeddings.token_type_embeddings.weight requires_grad= True
roberta.embeddings.LayerNorm.weight requires_grad= True
roberta.embeddings.LayerNorm.bias requires_grad= True
roberta.encoder.layer.0.attention.self.query.weight requires_grad= True
roberta.encoder.layer.0.attention.self.query.bias requires_grad= True
roberta.encoder.layer.0.attention.self.key.weight requires_grad= True
roberta.encoder.layer.0.attention.self.key.bias requires_grad= True
roberta.encoder.layer.0.attention.self.value.weight requires_grad= True
roberta.encoder.layer.0.attention.self.value.bias requires_grad= True
roberta.encoder.layer.0.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.0.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.0.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.0.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.0.output.dense.weight requires_grad= True
roberta.encoder.layer.0.output.dense.bias requires_grad= True
roberta.encoder.layer.0.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.0.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.1.attention.self.query.weight requires_grad= True
roberta.encoder.layer.1.attention.self.query.bias requires_grad= True
roberta.encoder.layer.1.attention.self.key.weight requires_grad= True
roberta.encoder.layer.1.attention.self.key.bias requires_grad= True
roberta.encoder.layer.1.attention.self.value.weight requires_grad= True
roberta.encoder.layer.1.attention.self.value.bias requires_grad= True
roberta.encoder.layer.1.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.1.attention.output.dense.bias requires_grad= True
```

roberta.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.1.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.1.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.1.output.dense.weight requires_grad= True
roberta.encoder.layer.1.output.dense.bias requires_grad= True
roberta.encoder.layer.1.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.1.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.2.attention.self.query.weight requires_grad= True
roberta.encoder.layer.2.attention.self.query.bias requires_grad= True
roberta.encoder.layer.2.attention.self.key.weight requires_grad= True
roberta.encoder.layer.2.attention.self.key.bias requires_grad= True
roberta.encoder.layer.2.attention.self.value.weight requires_grad= True
roberta.encoder.layer.2.attention.self.value.bias requires_grad= True
roberta.encoder.layer.2.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.2.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.2.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.2.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.2.output.dense.weight requires_grad= True
roberta.encoder.layer.2.output.dense.bias requires_grad= True
roberta.encoder.layer.2.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.2.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.3.attention.self.query.weight requires_grad= True
roberta.encoder.layer.3.attention.self.query.bias requires_grad= True
roberta.encoder.layer.3.attention.self.key.weight requires_grad= True
roberta.encoder.layer.3.attention.self.key.bias requires_grad= True
roberta.encoder.layer.3.attention.self.value.weight requires_grad= True
roberta.encoder.layer.3.attention.self.value.bias requires_grad= True
roberta.encoder.layer.3.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.3.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.3.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.3.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.3.output.dense.weight requires_grad= True
roberta.encoder.layer.3.output.dense.bias requires_grad= True
roberta.encoder.layer.3.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.3.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.4.attention.self.query.weight requires_grad= True
roberta.encoder.layer.4.attention.self.query.bias requires_grad= True
roberta.encoder.layer.4.attention.self.key.weight requires_grad= True
roberta.encoder.layer.4.attention.self.key.bias requires_grad= True
roberta.encoder.layer.4.attention.self.value.weight requires_grad= True
roberta.encoder.layer.4.attention.self.value.bias requires_grad= True
roberta.encoder.layer.4.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.4.attention.output.dense.bias requires_grad= True

roberta.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.4.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.4.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.4.output.dense.weight requires_grad= True
roberta.encoder.layer.4.output.dense.bias requires_grad= True
roberta.encoder.layer.4.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.4.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.5.attention.self.query.weight requires_grad= True
roberta.encoder.layer.5.attention.self.query.bias requires_grad= True
roberta.encoder.layer.5.attention.self.key.weight requires_grad= True
roberta.encoder.layer.5.attention.self.key.bias requires_grad= True
roberta.encoder.layer.5.attention.self.value.weight requires_grad= True
roberta.encoder.layer.5.attention.self.value.bias requires_grad= True
roberta.encoder.layer.5.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.5.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.5.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.5.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.5.output.dense.weight requires_grad= True
roberta.encoder.layer.5.output.dense.bias requires_grad= True
roberta.encoder.layer.5.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.5.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.6.attention.self.query.weight requires_grad= True
roberta.encoder.layer.6.attention.self.query.bias requires_grad= True
roberta.encoder.layer.6.attention.self.key.weight requires_grad= True
roberta.encoder.layer.6.attention.self.key.bias requires_grad= True
roberta.encoder.layer.6.attention.self.value.weight requires_grad= True
roberta.encoder.layer.6.attention.self.value.bias requires_grad= True
roberta.encoder.layer.6.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.6.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.6.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.6.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.6.output.dense.weight requires_grad= True
roberta.encoder.layer.6.output.dense.bias requires_grad= True
roberta.encoder.layer.6.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.6.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.7.attention.self.query.weight requires_grad= True
roberta.encoder.layer.7.attention.self.query.bias requires_grad= True
roberta.encoder.layer.7.attention.self.key.weight requires_grad= True
roberta.encoder.layer.7.attention.self.key.bias requires_grad= True
roberta.encoder.layer.7.attention.self.value.weight requires_grad= True
roberta.encoder.layer.7.attention.self.value.bias requires_grad= True
roberta.encoder.layer.7.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.7.attention.output.dense.bias requires_grad= True

roberta.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.7.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.7.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.7.output.dense.weight requires_grad= True
roberta.encoder.layer.7.output.dense.bias requires_grad= True
roberta.encoder.layer.7.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.7.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.8.attention.self.query.weight requires_grad= True
roberta.encoder.layer.8.attention.self.query.bias requires_grad= True
roberta.encoder.layer.8.attention.self.key.weight requires_grad= True
roberta.encoder.layer.8.attention.self.key.bias requires_grad= True
roberta.encoder.layer.8.attention.self.value.weight requires_grad= True
roberta.encoder.layer.8.attention.self.value.bias requires_grad= True
roberta.encoder.layer.8.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.8.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.8.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.8.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.8.output.dense.weight requires_grad= True
roberta.encoder.layer.8.output.dense.bias requires_grad= True
roberta.encoder.layer.8.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.8.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.9.attention.self.query.weight requires_grad= True
roberta.encoder.layer.9.attention.self.query.bias requires_grad= True
roberta.encoder.layer.9.attention.self.key.weight requires_grad= True
roberta.encoder.layer.9.attention.self.key.bias requires_grad= True
roberta.encoder.layer.9.attention.self.value.weight requires_grad= True
roberta.encoder.layer.9.attention.self.value.bias requires_grad= True
roberta.encoder.layer.9.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.9.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.9.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.9.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.9.output.dense.weight requires_grad= True
roberta.encoder.layer.9.output.dense.bias requires_grad= True
roberta.encoder.layer.9.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.9.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.10.attention.self.query.weight requires_grad= True
roberta.encoder.layer.10.attention.self.query.bias requires_grad= True
roberta.encoder.layer.10.attention.self.key.weight requires_grad= True
roberta.encoder.layer.10.attention.self.key.bias requires_grad= True
roberta.encoder.layer.10.attention.self.value.weight requires_grad= True
roberta.encoder.layer.10.attention.self.value.bias requires_grad= True
roberta.encoder.layer.10.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.10.attention.output.dense.bias requires_grad= True

roberta.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.10.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.10.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.10.output.dense.weight requires_grad= True
roberta.encoder.layer.10.output.dense.bias requires_grad= True
roberta.encoder.layer.10.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.10.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.11.attention.self.query.weight requires_grad= True
roberta.encoder.layer.11.attention.self.query.bias requires_grad= True
roberta.encoder.layer.11.attention.self.key.weight requires_grad= True
roberta.encoder.layer.11.attention.self.key.bias requires_grad= True
roberta.encoder.layer.11.attention.self.value.weight requires_grad= True
roberta.encoder.layer.11.attention.self.value.bias requires_grad= True
roberta.encoder.layer.11.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.11.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.11.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.11.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.11.output.dense.weight requires_grad= True
roberta.encoder.layer.11.output.dense.bias requires_grad= True
roberta.encoder.layer.11.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.11.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.12.attention.self.query.weight requires_grad= True
roberta.encoder.layer.12.attention.self.query.bias requires_grad= True
roberta.encoder.layer.12.attention.self.key.weight requires_grad= True
roberta.encoder.layer.12.attention.self.key.bias requires_grad= True
roberta.encoder.layer.12.attention.self.value.weight requires_grad= True
roberta.encoder.layer.12.attention.self.value.bias requires_grad= True
roberta.encoder.layer.12.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.12.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.12.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.12.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.12.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.12.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.12.output.dense.weight requires_grad= True
roberta.encoder.layer.12.output.dense.bias requires_grad= True
roberta.encoder.layer.12.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.12.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.13.attention.self.query.weight requires_grad= True
roberta.encoder.layer.13.attention.self.query.bias requires_grad= True
roberta.encoder.layer.13.attention.self.key.weight requires_grad= True
roberta.encoder.layer.13.attention.self.key.bias requires_grad= True
roberta.encoder.layer.13.attention.self.value.weight requires_grad= True
roberta.encoder.layer.13.attention.self.value.bias requires_grad= True
roberta.encoder.layer.13.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.13.attention.output.dense.bias requires_grad= True

roberta.encoder.layer.13.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.13.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.13.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.13.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.13.output.dense.weight requires_grad= True
roberta.encoder.layer.13.output.dense.bias requires_grad= True
roberta.encoder.layer.13.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.13.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.14.attention.self.query.weight requires_grad= True
roberta.encoder.layer.14.attention.self.query.bias requires_grad= True
roberta.encoder.layer.14.attention.self.key.weight requires_grad= True
roberta.encoder.layer.14.attention.self.key.bias requires_grad= True
roberta.encoder.layer.14.attention.self.value.weight requires_grad= True
roberta.encoder.layer.14.attention.self.value.bias requires_grad= True
roberta.encoder.layer.14.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.14.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.14.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.14.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.14.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.14.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.14.output.dense.weight requires_grad= True
roberta.encoder.layer.14.output.dense.bias requires_grad= True
roberta.encoder.layer.14.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.14.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.15.attention.self.query.weight requires_grad= True
roberta.encoder.layer.15.attention.self.query.bias requires_grad= True
roberta.encoder.layer.15.attention.self.key.weight requires_grad= True
roberta.encoder.layer.15.attention.self.key.bias requires_grad= True
roberta.encoder.layer.15.attention.self.value.weight requires_grad= True
roberta.encoder.layer.15.attention.self.value.bias requires_grad= True
roberta.encoder.layer.15.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.15.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.15.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.15.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.15.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.15.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.15.output.dense.weight requires_grad= True
roberta.encoder.layer.15.output.dense.bias requires_grad= True
roberta.encoder.layer.15.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.15.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.16.attention.self.query.weight requires_grad= True
roberta.encoder.layer.16.attention.self.query.bias requires_grad= True
roberta.encoder.layer.16.attention.self.key.weight requires_grad= True
roberta.encoder.layer.16.attention.self.key.bias requires_grad= True
roberta.encoder.layer.16.attention.self.value.weight requires_grad= True
roberta.encoder.layer.16.attention.self.value.bias requires_grad= True
roberta.encoder.layer.16.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.16.attention.output.dense.bias requires_grad= True

```
roberta.encoder.layer.16.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.16.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.16.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.16.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.16.output.dense.weight requires_grad= True
roberta.encoder.layer.16.output.dense.bias requires_grad= True
roberta.encoder.layer.16.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.16.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.17.attention.self.query.weight requires_grad= True
roberta.encoder.layer.17.attention.self.query.bias requires_grad= True
roberta.encoder.layer.17.attention.self.key.weight requires_grad= True
roberta.encoder.layer.17.attention.self.key.bias requires_grad= True
roberta.encoder.layer.17.attention.self.value.weight requires_grad= True
roberta.encoder.layer.17.attention.self.value.bias requires_grad= True
roberta.encoder.layer.17.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.17.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.17.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.17.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.17.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.17.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.17.output.dense.weight requires_grad= True
roberta.encoder.layer.17.output.dense.bias requires_grad= True
roberta.encoder.layer.17.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.17.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.18.attention.self.query.weight requires_grad= True
roberta.encoder.layer.18.attention.self.query.bias requires_grad= True
roberta.encoder.layer.18.attention.self.key.weight requires_grad= True
roberta.encoder.layer.18.attention.self.key.bias requires_grad= True
roberta.encoder.layer.18.attention.self.value.weight requires_grad= True
roberta.encoder.layer.18.attention.self.value.bias requires_grad= True
roberta.encoder.layer.18.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.18.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.18.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.18.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.18.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.18.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.18.output.dense.weight requires_grad= True
roberta.encoder.layer.18.output.dense.bias requires_grad= True
roberta.encoder.layer.18.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.18.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.19.attention.self.query.weight requires_grad= True
roberta.encoder.layer.19.attention.self.query.bias requires_grad= True
roberta.encoder.layer.19.attention.self.key.weight requires_grad= True
roberta.encoder.layer.19.attention.self.key.bias requires_grad= True
roberta.encoder.layer.19.attention.self.value.weight requires_grad= True
roberta.encoder.layer.19.attention.self.value.bias requires_grad= True
roberta.encoder.layer.19.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.19.attention.output.dense.bias requires_grad= True
```

```
roberta.encoder.layer.19.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.19.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.19.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.19.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.19.output.dense.weight requires_grad= True
roberta.encoder.layer.19.output.dense.bias requires_grad= True
roberta.encoder.layer.19.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.19.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.20.attention.self.query.weight requires_grad= True
roberta.encoder.layer.20.attention.self.query.bias requires_grad= True
roberta.encoder.layer.20.attention.self.key.weight requires_grad= True
roberta.encoder.layer.20.attention.self.key.bias requires_grad= True
roberta.encoder.layer.20.attention.self.value.weight requires_grad= True
roberta.encoder.layer.20.attention.self.value.bias requires_grad= True
roberta.encoder.layer.20.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.20.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.20.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.20.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.20.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.20.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.20.output.dense.weight requires_grad= True
roberta.encoder.layer.20.output.dense.bias requires_grad= True
roberta.encoder.layer.20.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.20.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.21.attention.self.query.weight requires_grad= True
roberta.encoder.layer.21.attention.self.query.bias requires_grad= True
roberta.encoder.layer.21.attention.self.key.weight requires_grad= True
roberta.encoder.layer.21.attention.self.key.bias requires_grad= True
roberta.encoder.layer.21.attention.self.value.weight requires_grad= True
roberta.encoder.layer.21.attention.self.value.bias requires_grad= True
roberta.encoder.layer.21.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.21.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.21.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.21.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.21.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.21.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.21.output.dense.weight requires_grad= True
roberta.encoder.layer.21.output.dense.bias requires_grad= True
roberta.encoder.layer.21.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.21.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.22.attention.self.query.weight requires_grad= True
roberta.encoder.layer.22.attention.self.query.bias requires_grad= True
roberta.encoder.layer.22.attention.self.key.weight requires_grad= True
roberta.encoder.layer.22.attention.self.key.bias requires_grad= True
roberta.encoder.layer.22.attention.self.value.weight requires_grad= True
roberta.encoder.layer.22.attention.self.value.bias requires_grad= True
roberta.encoder.layer.22.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.22.attention.output.dense.bias requires_grad= True
```

```
roberta.encoder.layer.22.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.22.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.22.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.22.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.22.output.dense.weight requires_grad= True
roberta.encoder.layer.22.output.dense.bias requires_grad= True
roberta.encoder.layer.22.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.22.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.23.attention.self.query.weight requires_grad= True
roberta.encoder.layer.23.attention.self.query.bias requires_grad= True
roberta.encoder.layer.23.attention.self.key.weight requires_grad= True
roberta.encoder.layer.23.attention.self.key.bias requires_grad= True
roberta.encoder.layer.23.attention.self.value.weight requires_grad= True
roberta.encoder.layer.23.attention.self.value.bias requires_grad= True
roberta.encoder.layer.23.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.23.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.23.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.23.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.23.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.23.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.23.output.dense.weight requires_grad= True
roberta.encoder.layer.23.output.dense.bias requires_grad= True
roberta.encoder.layer.23.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.23.output.LayerNorm.bias requires_grad= True
classifier.dense.weight requires_grad= True
classifier.dense.bias requires_grad= True
classifier.out_proj.weight requires_grad= True
classifier.out_proj.bias requires_grad= True
```

[ ]:
```python
# Inspect the attention_mask tensor for the first few samples
for i in range(5):
    print(train_data_hf[i]['attention_mask'])
```

```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
```

[ ]:
```python
########################################
layers_to_unfreeze = [
    "roberta.encoder.layer.23.attention.self.query.weight",
    "roberta.encoder.layer.23.attention.self.query.bias",
    "roberta.encoder.layer.23.attention.self.key.weight",
    "roberta.encoder.layer.23.attention.self.key.bias",
    "roberta.encoder.layer.23.attention.self.value.weight",
    "roberta.encoder.layer.23.attention.self.value.bias",
    "roberta.encoder.layer.23.attention.output.dense.weight",
    "roberta.encoder.layer.23.attention.output.dense.bias",
    "roberta.encoder.layer.23.attention.output.LayerNorm.weight",
    "roberta.encoder.layer.23.attention.output.LayerNorm.bias",
    "roberta.encoder.layer.23.intermediate.dense.weight",
    "roberta.encoder.layer.23.intermediate.dense.bias",
    "roberta.encoder.layer.23.output.dense.weight",
    "roberta.encoder.layer.23.output.dense.bias",
    "roberta.encoder.layer.23.output.LayerNorm.weight",
    "roberta.encoder.layer.23.output.LayerNorm.bias",
    "classifier.dense.weight",
    "classifier.dense.bias",
    "classifier.out_proj.weight",
    "classifier.out_proj.bias",
]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("=============")
########################################
print("Experiment configuration used with this experiment:")
```

```python
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
########################################
print("=============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```

```
RobertaConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "RobertaForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-05,
  "max_position_embeddings": 514,
  "model_type": "roberta",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "type_vocab_size": 1,
  "use_cache": true,
  "vocab_size": 50265
}

=============
num_parameters: 355361794
num_trainable_parameters: 13647874
=============
Experiment configuration used with this experiment:
model used: roberta-large
learning rate used: 5e-06
```

```
number of epochs: 1
maximum sequence length: 128
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
=============
num_trainable_parameters: 13647874
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
roberta.embeddings.word_embeddings.weight requires_grad= False
roberta.embeddings.position_embeddings.weight requires_grad= False
roberta.embeddings.token_type_embeddings.weight requires_grad= False
roberta.embeddings.LayerNorm.weight requires_grad= False
roberta.embeddings.LayerNorm.bias requires_grad= False
roberta.encoder.layer.0.attention.self.query.weight requires_grad= False
roberta.encoder.layer.0.attention.self.query.bias requires_grad= False
roberta.encoder.layer.0.attention.self.key.weight requires_grad= False
roberta.encoder.layer.0.attention.self.key.bias requires_grad= False
roberta.encoder.layer.0.attention.self.value.weight requires_grad= False
roberta.encoder.layer.0.attention.self.value.bias requires_grad= False
roberta.encoder.layer.0.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.0.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.0.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.0.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.0.output.dense.weight requires_grad= False
roberta.encoder.layer.0.output.dense.bias requires_grad= False
roberta.encoder.layer.0.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.0.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.1.attention.self.query.weight requires_grad= False
roberta.encoder.layer.1.attention.self.query.bias requires_grad= False
roberta.encoder.layer.1.attention.self.key.weight requires_grad= False
roberta.encoder.layer.1.attention.self.key.bias requires_grad= False
roberta.encoder.layer.1.attention.self.value.weight requires_grad= False
roberta.encoder.layer.1.attention.self.value.bias requires_grad= False
roberta.encoder.layer.1.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.1.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.1.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.1.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.1.output.dense.weight requires_grad= False
roberta.encoder.layer.1.output.dense.bias requires_grad= False
```

```
roberta.encoder.layer.1.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.1.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.2.attention.self.query.weight requires_grad= False
roberta.encoder.layer.2.attention.self.query.bias requires_grad= False
roberta.encoder.layer.2.attention.self.key.weight requires_grad= False
roberta.encoder.layer.2.attention.self.key.bias requires_grad= False
roberta.encoder.layer.2.attention.self.value.weight requires_grad= False
roberta.encoder.layer.2.attention.self.value.bias requires_grad= False
roberta.encoder.layer.2.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.2.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.2.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.2.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.2.output.dense.weight requires_grad= False
roberta.encoder.layer.2.output.dense.bias requires_grad= False
roberta.encoder.layer.2.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.2.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.3.attention.self.query.weight requires_grad= False
roberta.encoder.layer.3.attention.self.query.bias requires_grad= False
roberta.encoder.layer.3.attention.self.key.weight requires_grad= False
roberta.encoder.layer.3.attention.self.key.bias requires_grad= False
roberta.encoder.layer.3.attention.self.value.weight requires_grad= False
roberta.encoder.layer.3.attention.self.value.bias requires_grad= False
roberta.encoder.layer.3.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.3.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.3.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.3.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.3.output.dense.weight requires_grad= False
roberta.encoder.layer.3.output.dense.bias requires_grad= False
roberta.encoder.layer.3.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.3.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.4.attention.self.query.weight requires_grad= False
roberta.encoder.layer.4.attention.self.query.bias requires_grad= False
roberta.encoder.layer.4.attention.self.key.weight requires_grad= False
roberta.encoder.layer.4.attention.self.key.bias requires_grad= False
roberta.encoder.layer.4.attention.self.value.weight requires_grad= False
roberta.encoder.layer.4.attention.self.value.bias requires_grad= False
roberta.encoder.layer.4.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.4.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.4.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.4.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.4.output.dense.weight requires_grad= False
roberta.encoder.layer.4.output.dense.bias requires_grad= False
```

```
roberta.encoder.layer.4.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.4.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.5.attention.self.query.weight requires_grad= False
roberta.encoder.layer.5.attention.self.query.bias requires_grad= False
roberta.encoder.layer.5.attention.self.key.weight requires_grad= False
roberta.encoder.layer.5.attention.self.key.bias requires_grad= False
roberta.encoder.layer.5.attention.self.value.weight requires_grad= False
roberta.encoder.layer.5.attention.self.value.bias requires_grad= False
roberta.encoder.layer.5.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.5.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.5.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.5.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.5.output.dense.weight requires_grad= False
roberta.encoder.layer.5.output.dense.bias requires_grad= False
roberta.encoder.layer.5.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.5.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.6.attention.self.query.weight requires_grad= False
roberta.encoder.layer.6.attention.self.query.bias requires_grad= False
roberta.encoder.layer.6.attention.self.key.weight requires_grad= False
roberta.encoder.layer.6.attention.self.key.bias requires_grad= False
roberta.encoder.layer.6.attention.self.value.weight requires_grad= False
roberta.encoder.layer.6.attention.self.value.bias requires_grad= False
roberta.encoder.layer.6.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.6.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.6.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.6.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.6.output.dense.weight requires_grad= False
roberta.encoder.layer.6.output.dense.bias requires_grad= False
roberta.encoder.layer.6.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.6.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.7.attention.self.query.weight requires_grad= False
roberta.encoder.layer.7.attention.self.query.bias requires_grad= False
roberta.encoder.layer.7.attention.self.key.weight requires_grad= False
roberta.encoder.layer.7.attention.self.key.bias requires_grad= False
roberta.encoder.layer.7.attention.self.value.weight requires_grad= False
roberta.encoder.layer.7.attention.self.value.bias requires_grad= False
roberta.encoder.layer.7.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.7.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.7.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.7.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.7.output.dense.weight requires_grad= False
roberta.encoder.layer.7.output.dense.bias requires_grad= False
```

```
roberta.encoder.layer.7.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.7.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.8.attention.self.query.weight requires_grad= False
roberta.encoder.layer.8.attention.self.query.bias requires_grad= False
roberta.encoder.layer.8.attention.self.key.weight requires_grad= False
roberta.encoder.layer.8.attention.self.key.bias requires_grad= False
roberta.encoder.layer.8.attention.self.value.weight requires_grad= False
roberta.encoder.layer.8.attention.self.value.bias requires_grad= False
roberta.encoder.layer.8.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.8.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.8.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.8.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.8.output.dense.weight requires_grad= False
roberta.encoder.layer.8.output.dense.bias requires_grad= False
roberta.encoder.layer.8.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.8.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.9.attention.self.query.weight requires_grad= False
roberta.encoder.layer.9.attention.self.query.bias requires_grad= False
roberta.encoder.layer.9.attention.self.key.weight requires_grad= False
roberta.encoder.layer.9.attention.self.key.bias requires_grad= False
roberta.encoder.layer.9.attention.self.value.weight requires_grad= False
roberta.encoder.layer.9.attention.self.value.bias requires_grad= False
roberta.encoder.layer.9.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.9.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.9.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.9.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.9.output.dense.weight requires_grad= False
roberta.encoder.layer.9.output.dense.bias requires_grad= False
roberta.encoder.layer.9.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.9.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.10.attention.self.query.weight requires_grad= False
roberta.encoder.layer.10.attention.self.query.bias requires_grad= False
roberta.encoder.layer.10.attention.self.key.weight requires_grad= False
roberta.encoder.layer.10.attention.self.key.bias requires_grad= False
roberta.encoder.layer.10.attention.self.value.weight requires_grad= False
roberta.encoder.layer.10.attention.self.value.bias requires_grad= False
roberta.encoder.layer.10.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.10.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.10.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.10.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.10.output.dense.weight requires_grad= False
roberta.encoder.layer.10.output.dense.bias requires_grad= False
```

```
roberta.encoder.layer.10.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.10.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.11.attention.self.query.weight requires_grad= False
roberta.encoder.layer.11.attention.self.query.bias requires_grad= False
roberta.encoder.layer.11.attention.self.key.weight requires_grad= False
roberta.encoder.layer.11.attention.self.key.bias requires_grad= False
roberta.encoder.layer.11.attention.self.value.weight requires_grad= False
roberta.encoder.layer.11.attention.self.value.bias requires_grad= False
roberta.encoder.layer.11.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.11.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.11.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.11.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.11.output.dense.weight requires_grad= False
roberta.encoder.layer.11.output.dense.bias requires_grad= False
roberta.encoder.layer.11.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.11.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.12.attention.self.query.weight requires_grad= False
roberta.encoder.layer.12.attention.self.query.bias requires_grad= False
roberta.encoder.layer.12.attention.self.key.weight requires_grad= False
roberta.encoder.layer.12.attention.self.key.bias requires_grad= False
roberta.encoder.layer.12.attention.self.value.weight requires_grad= False
roberta.encoder.layer.12.attention.self.value.bias requires_grad= False
roberta.encoder.layer.12.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.12.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.12.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.12.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.12.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.12.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.12.output.dense.weight requires_grad= False
roberta.encoder.layer.12.output.dense.bias requires_grad= False
roberta.encoder.layer.12.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.12.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.13.attention.self.query.weight requires_grad= False
roberta.encoder.layer.13.attention.self.query.bias requires_grad= False
roberta.encoder.layer.13.attention.self.key.weight requires_grad= False
roberta.encoder.layer.13.attention.self.key.bias requires_grad= False
roberta.encoder.layer.13.attention.self.value.weight requires_grad= False
roberta.encoder.layer.13.attention.self.value.bias requires_grad= False
roberta.encoder.layer.13.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.13.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.13.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.13.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.13.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.13.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.13.output.dense.weight requires_grad= False
roberta.encoder.layer.13.output.dense.bias requires_grad= False
```

roberta.encoder.layer.13.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.13.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.14.attention.self.query.weight requires_grad= False
roberta.encoder.layer.14.attention.self.query.bias requires_grad= False
roberta.encoder.layer.14.attention.self.key.weight requires_grad= False
roberta.encoder.layer.14.attention.self.key.bias requires_grad= False
roberta.encoder.layer.14.attention.self.value.weight requires_grad= False
roberta.encoder.layer.14.attention.self.value.bias requires_grad= False
roberta.encoder.layer.14.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.14.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.14.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.14.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.14.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.14.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.14.output.dense.weight requires_grad= False
roberta.encoder.layer.14.output.dense.bias requires_grad= False
roberta.encoder.layer.14.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.14.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.15.attention.self.query.weight requires_grad= False
roberta.encoder.layer.15.attention.self.query.bias requires_grad= False
roberta.encoder.layer.15.attention.self.key.weight requires_grad= False
roberta.encoder.layer.15.attention.self.key.bias requires_grad= False
roberta.encoder.layer.15.attention.self.value.weight requires_grad= False
roberta.encoder.layer.15.attention.self.value.bias requires_grad= False
roberta.encoder.layer.15.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.15.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.15.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.15.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.15.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.15.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.15.output.dense.weight requires_grad= False
roberta.encoder.layer.15.output.dense.bias requires_grad= False
roberta.encoder.layer.15.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.15.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.16.attention.self.query.weight requires_grad= False
roberta.encoder.layer.16.attention.self.query.bias requires_grad= False
roberta.encoder.layer.16.attention.self.key.weight requires_grad= False
roberta.encoder.layer.16.attention.self.key.bias requires_grad= False
roberta.encoder.layer.16.attention.self.value.weight requires_grad= False
roberta.encoder.layer.16.attention.self.value.bias requires_grad= False
roberta.encoder.layer.16.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.16.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.16.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.16.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.16.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.16.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.16.output.dense.weight requires_grad= False
roberta.encoder.layer.16.output.dense.bias requires_grad= False

roberta.encoder.layer.16.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.16.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.17.attention.self.query.weight requires_grad= False
roberta.encoder.layer.17.attention.self.query.bias requires_grad= False
roberta.encoder.layer.17.attention.self.key.weight requires_grad= False
roberta.encoder.layer.17.attention.self.key.bias requires_grad= False
roberta.encoder.layer.17.attention.self.value.weight requires_grad= False
roberta.encoder.layer.17.attention.self.value.bias requires_grad= False
roberta.encoder.layer.17.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.17.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.17.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.17.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.17.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.17.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.17.output.dense.weight requires_grad= False
roberta.encoder.layer.17.output.dense.bias requires_grad= False
roberta.encoder.layer.17.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.17.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.18.attention.self.query.weight requires_grad= False
roberta.encoder.layer.18.attention.self.query.bias requires_grad= False
roberta.encoder.layer.18.attention.self.key.weight requires_grad= False
roberta.encoder.layer.18.attention.self.key.bias requires_grad= False
roberta.encoder.layer.18.attention.self.value.weight requires_grad= False
roberta.encoder.layer.18.attention.self.value.bias requires_grad= False
roberta.encoder.layer.18.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.18.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.18.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.18.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.18.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.18.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.18.output.dense.weight requires_grad= False
roberta.encoder.layer.18.output.dense.bias requires_grad= False
roberta.encoder.layer.18.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.18.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.19.attention.self.query.weight requires_grad= False
roberta.encoder.layer.19.attention.self.query.bias requires_grad= False
roberta.encoder.layer.19.attention.self.key.weight requires_grad= False
roberta.encoder.layer.19.attention.self.key.bias requires_grad= False
roberta.encoder.layer.19.attention.self.value.weight requires_grad= False
roberta.encoder.layer.19.attention.self.value.bias requires_grad= False
roberta.encoder.layer.19.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.19.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.19.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.19.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.19.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.19.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.19.output.dense.weight requires_grad= False
roberta.encoder.layer.19.output.dense.bias requires_grad= False

```
roberta.encoder.layer.19.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.19.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.20.attention.self.query.weight requires_grad= False
roberta.encoder.layer.20.attention.self.query.bias requires_grad= False
roberta.encoder.layer.20.attention.self.key.weight requires_grad= False
roberta.encoder.layer.20.attention.self.key.bias requires_grad= False
roberta.encoder.layer.20.attention.self.value.weight requires_grad= False
roberta.encoder.layer.20.attention.self.value.bias requires_grad= False
roberta.encoder.layer.20.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.20.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.20.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.20.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.20.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.20.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.20.output.dense.weight requires_grad= False
roberta.encoder.layer.20.output.dense.bias requires_grad= False
roberta.encoder.layer.20.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.20.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.21.attention.self.query.weight requires_grad= False
roberta.encoder.layer.21.attention.self.query.bias requires_grad= False
roberta.encoder.layer.21.attention.self.key.weight requires_grad= False
roberta.encoder.layer.21.attention.self.key.bias requires_grad= False
roberta.encoder.layer.21.attention.self.value.weight requires_grad= False
roberta.encoder.layer.21.attention.self.value.bias requires_grad= False
roberta.encoder.layer.21.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.21.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.21.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.21.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.21.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.21.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.21.output.dense.weight requires_grad= False
roberta.encoder.layer.21.output.dense.bias requires_grad= False
roberta.encoder.layer.21.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.21.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.22.attention.self.query.weight requires_grad= False
roberta.encoder.layer.22.attention.self.query.bias requires_grad= False
roberta.encoder.layer.22.attention.self.key.weight requires_grad= False
roberta.encoder.layer.22.attention.self.key.bias requires_grad= False
roberta.encoder.layer.22.attention.self.value.weight requires_grad= False
roberta.encoder.layer.22.attention.self.value.bias requires_grad= False
roberta.encoder.layer.22.attention.output.dense.weight requires_grad= False
roberta.encoder.layer.22.attention.output.dense.bias requires_grad= False
roberta.encoder.layer.22.attention.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.22.attention.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.22.intermediate.dense.weight requires_grad= False
roberta.encoder.layer.22.intermediate.dense.bias requires_grad= False
roberta.encoder.layer.22.output.dense.weight requires_grad= False
roberta.encoder.layer.22.output.dense.bias requires_grad= False
```

```
roberta.encoder.layer.22.output.LayerNorm.weight requires_grad= False
roberta.encoder.layer.22.output.LayerNorm.bias requires_grad= False
roberta.encoder.layer.23.attention.self.query.weight requires_grad= True
roberta.encoder.layer.23.attention.self.query.bias requires_grad= True
roberta.encoder.layer.23.attention.self.key.weight requires_grad= True
roberta.encoder.layer.23.attention.self.key.bias requires_grad= True
roberta.encoder.layer.23.attention.self.value.weight requires_grad= True
roberta.encoder.layer.23.attention.self.value.bias requires_grad= True
roberta.encoder.layer.23.attention.output.dense.weight requires_grad= True
roberta.encoder.layer.23.attention.output.dense.bias requires_grad= True
roberta.encoder.layer.23.attention.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.23.attention.output.LayerNorm.bias requires_grad= True
roberta.encoder.layer.23.intermediate.dense.weight requires_grad= True
roberta.encoder.layer.23.intermediate.dense.bias requires_grad= True
roberta.encoder.layer.23.output.dense.weight requires_grad= True
roberta.encoder.layer.23.output.dense.bias requires_grad= True
roberta.encoder.layer.23.output.LayerNorm.weight requires_grad= True
roberta.encoder.layer.23.output.LayerNorm.bias requires_grad= True
classifier.dense.weight requires_grad= True
classifier.dense.bias requires_grad= True
classifier.out_proj.weight requires_grad= True
classifier.out_proj.bias requires_grad= True
```

[ ]:
```
model.resize_token_embeddings(len(tokenizer))
```

[ ]:
```
Embedding(50265, 1024, padding_idx=1)
```

[ ]:
```
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
    val_dataset = val_data_hf,
    output_dir = dir_results,
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-20-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
```

```
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 0.6940522193908691, 'eval_accuracy':
0.45368171021377673, 'eval_precision': 0.4538834951456311, 'eval_recall':
0.9739583333333334, 'eval_f1': 0.6192052980132451, 'eval_runtime': 3.0907,
'eval_samples_per_second': 136.214, 'eval_steps_per_second': 1.294, 'epoch':
1.0}
Test metrics: {'eval_loss': 0.6936661005020142, 'eval_accuracy':
0.4841875681570338, 'eval_precision': 0.4819004524886878, 'eval_recall':
0.9659863945578231, 'eval_f1': 0.6430188679245283, 'eval_runtime': 5.19,
'eval_samples_per_second': 176.686, 'eval_steps_per_second': 1.541, 'epoch':
1.0}
```

[ ]:
```
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
  ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to:
/content/drive/MyDrive/266-final/models/single_roberta-
large_binary_complexity_20250411_121257

<IPython.core.display.HTML object>

EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```

0.2.7 snc answerdotai/ModernBERT-base regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[ ]:
```python
# Define Experiment Parameters
# named_model = "bert-base-cased"
# named_model = "roberta-base"
# named_model = "bert-large-cased"
# named_model = "roberta-large"
named_model = "answerdotai/ModernBERT-base" # modern bert
#############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
num_epochs = 1
##########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
#############
y_col = "binary_complexity"
# y_col = "complexity"
#############
x_task = "single"
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
##########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
```

```
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
##########################################
# custom_config = BertConfig.from_pretrained("roberta-base")
# custom_config.hidden_act = "gelu"  # alts: "relu" "silu"
# custom_config.attention_probs_dropout_prob = 0.1
# custom_config.hidden_dropout_prob = 0.1
# custom_config.gradient_checkpointing = False
##########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="answerdotai/ModernBERT-base",
    local_model_path=None,
    config=None)
############
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
#     local_model_path="...CONFIGURE_PATH...",
#     config=custom_config)
print("=============")
print(named_model, ":")
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters at load:", model.
 ↪num_parameters(only_trainable=True))
print("=============")
print("model lineage:", MODEL_LINEAGE)
print("=============")
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/421 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/917 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([    0,  1708,     7,     5,  7250,
9,  6987,  2681,    37,   851,
         4146,     6,   142,     5,   544,     9,     5, 13488, 14574,     7,
          106,   131,    51,  2584,    24,    15,    49, 10762,     4,     2,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
            1,     1,     1,     1,     1,     1,     1,     1]),
'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])}
Loading from Hugging Face model: answerdotai/ModernBERT-base

tokenizer_config.json:   0%|          | 0.00/20.8k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/2.13M [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/694 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/1.19k [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/599M [00:00<?, ?B/s]

Some weights of ModernBertForSequenceClassification were not initialized from
the model checkpoint at answerdotai/ModernBERT-base and are newly initialized:
['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

=============
answerdotai/ModernBERT-base :
=============
num_parameters: 149606402
num_trainable_parameters at load: 149606402
=============
model lineage: {'type': 'huggingface_hub', 'path': 'answerdotai/ModernBERT-
base', 'timestamp': '2025-04-11 12:13:53'}
=============
```

[ ]:
```
print(model)
```

```
ModernBertForSequenceClassification(
  (model): ModernBertModel(
    (embeddings): ModernBertEmbeddings(
      (tok_embeddings): Embedding(50368, 768, padding_idx=50283)
      (norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (drop): Dropout(p=0.0, inplace=False)
    )
    (layers): ModuleList(
      (0): ModernBertEncoderLayer(
        (attn_norm): Identity()
        (attn): ModernBertAttention(
          (Wqkv): Linear(in_features=768, out_features=2304, bias=False)
          (rotary_emb): ModernBertRotaryEmbedding()
          (Wo): Linear(in_features=768, out_features=768, bias=False)
          (out_drop): Identity()
        )
        (mlp_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (mlp): ModernBertMLP(
          (Wi): Linear(in_features=768, out_features=2304, bias=False)
          (act): GELUActivation()
          (drop): Dropout(p=0.0, inplace=False)
          (Wo): Linear(in_features=1152, out_features=768, bias=False)
        )
      )
      (1-21): 21 x ModernBertEncoderLayer(
        (attn_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (attn): ModernBertAttention(
          (Wqkv): Linear(in_features=768, out_features=2304, bias=False)
          (rotary_emb): ModernBertRotaryEmbedding()
          (Wo): Linear(in_features=768, out_features=768, bias=False)
          (out_drop): Identity()
        )
        (mlp_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (mlp): ModernBertMLP(
          (Wi): Linear(in_features=768, out_features=2304, bias=False)
          (act): GELUActivation()
          (drop): Dropout(p=0.0, inplace=False)
          (Wo): Linear(in_features=1152, out_features=768, bias=False)
        )
      )
    )
    (final_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
  )
  (head): ModernBertPredictionHead(
    (dense): Linear(in_features=768, out_features=768, bias=False)
    (act): GELUActivation()
```

```
    (norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
  )
  (drop): Dropout(p=0.0, inplace=False)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
model.embeddings.tok_embeddings.weight requires_grad= True
model.embeddings.norm.weight requires_grad= True
model.layers.0.attn.Wqkv.weight requires_grad= True
model.layers.0.attn.Wo.weight requires_grad= True
model.layers.0.mlp_norm.weight requires_grad= True
model.layers.0.mlp.Wi.weight requires_grad= True
model.layers.0.mlp.Wo.weight requires_grad= True
model.layers.1.attn_norm.weight requires_grad= True
model.layers.1.attn.Wqkv.weight requires_grad= True
model.layers.1.attn.Wo.weight requires_grad= True
model.layers.1.mlp_norm.weight requires_grad= True
model.layers.1.mlp.Wi.weight requires_grad= True
model.layers.1.mlp.Wo.weight requires_grad= True
model.layers.2.attn_norm.weight requires_grad= True
model.layers.2.attn.Wqkv.weight requires_grad= True
model.layers.2.attn.Wo.weight requires_grad= True
model.layers.2.mlp_norm.weight requires_grad= True
model.layers.2.mlp.Wi.weight requires_grad= True
model.layers.2.mlp.Wo.weight requires_grad= True
model.layers.3.attn_norm.weight requires_grad= True
model.layers.3.attn.Wqkv.weight requires_grad= True
model.layers.3.attn.Wo.weight requires_grad= True
model.layers.3.mlp_norm.weight requires_grad= True
model.layers.3.mlp.Wi.weight requires_grad= True
model.layers.3.mlp.Wo.weight requires_grad= True
model.layers.4.attn_norm.weight requires_grad= True
model.layers.4.attn.Wqkv.weight requires_grad= True
model.layers.4.attn.Wo.weight requires_grad= True
model.layers.4.mlp_norm.weight requires_grad= True
model.layers.4.mlp.Wi.weight requires_grad= True
model.layers.4.mlp.Wo.weight requires_grad= True
model.layers.5.attn_norm.weight requires_grad= True
model.layers.5.attn.Wqkv.weight requires_grad= True
model.layers.5.attn.Wo.weight requires_grad= True
model.layers.5.mlp_norm.weight requires_grad= True
model.layers.5.mlp.Wi.weight requires_grad= True
model.layers.5.mlp.Wo.weight requires_grad= True
model.layers.6.attn_norm.weight requires_grad= True
model.layers.6.attn.Wqkv.weight requires_grad= True
```

```
model.layers.6.attn.Wo.weight requires_grad= True
model.layers.6.mlp_norm.weight requires_grad= True
model.layers.6.mlp.Wi.weight requires_grad= True
model.layers.6.mlp.Wo.weight requires_grad= True
model.layers.7.attn_norm.weight requires_grad= True
model.layers.7.attn.Wqkv.weight requires_grad= True
model.layers.7.attn.Wo.weight requires_grad= True
model.layers.7.mlp_norm.weight requires_grad= True
model.layers.7.mlp.Wi.weight requires_grad= True
model.layers.7.mlp.Wo.weight requires_grad= True
model.layers.8.attn_norm.weight requires_grad= True
model.layers.8.attn.Wqkv.weight requires_grad= True
model.layers.8.attn.Wo.weight requires_grad= True
model.layers.8.mlp_norm.weight requires_grad= True
model.layers.8.mlp.Wi.weight requires_grad= True
model.layers.8.mlp.Wo.weight requires_grad= True
model.layers.9.attn_norm.weight requires_grad= True
model.layers.9.attn.Wqkv.weight requires_grad= True
model.layers.9.attn.Wo.weight requires_grad= True
model.layers.9.mlp_norm.weight requires_grad= True
model.layers.9.mlp.Wi.weight requires_grad= True
model.layers.9.mlp.Wo.weight requires_grad= True
model.layers.10.attn_norm.weight requires_grad= True
model.layers.10.attn.Wqkv.weight requires_grad= True
model.layers.10.attn.Wo.weight requires_grad= True
model.layers.10.mlp_norm.weight requires_grad= True
model.layers.10.mlp.Wi.weight requires_grad= True
model.layers.10.mlp.Wo.weight requires_grad= True
model.layers.11.attn_norm.weight requires_grad= True
model.layers.11.attn.Wqkv.weight requires_grad= True
model.layers.11.attn.Wo.weight requires_grad= True
model.layers.11.mlp_norm.weight requires_grad= True
model.layers.11.mlp.Wi.weight requires_grad= True
model.layers.11.mlp.Wo.weight requires_grad= True
model.layers.12.attn_norm.weight requires_grad= True
model.layers.12.attn.Wqkv.weight requires_grad= True
model.layers.12.attn.Wo.weight requires_grad= True
model.layers.12.mlp_norm.weight requires_grad= True
model.layers.12.mlp.Wi.weight requires_grad= True
model.layers.12.mlp.Wo.weight requires_grad= True
model.layers.13.attn_norm.weight requires_grad= True
model.layers.13.attn.Wqkv.weight requires_grad= True
model.layers.13.attn.Wo.weight requires_grad= True
model.layers.13.mlp_norm.weight requires_grad= True
model.layers.13.mlp.Wi.weight requires_grad= True
model.layers.13.mlp.Wo.weight requires_grad= True
model.layers.14.attn_norm.weight requires_grad= True
model.layers.14.attn.Wqkv.weight requires_grad= True
```

```
model.layers.14.attn.Wo.weight requires_grad= True
model.layers.14.mlp_norm.weight requires_grad= True
model.layers.14.mlp.Wi.weight requires_grad= True
model.layers.14.mlp.Wo.weight requires_grad= True
model.layers.15.attn_norm.weight requires_grad= True
model.layers.15.attn.Wqkv.weight requires_grad= True
model.layers.15.attn.Wo.weight requires_grad= True
model.layers.15.mlp_norm.weight requires_grad= True
model.layers.15.mlp.Wi.weight requires_grad= True
model.layers.15.mlp.Wo.weight requires_grad= True
model.layers.16.attn_norm.weight requires_grad= True
model.layers.16.attn.Wqkv.weight requires_grad= True
model.layers.16.attn.Wo.weight requires_grad= True
model.layers.16.mlp_norm.weight requires_grad= True
model.layers.16.mlp.Wi.weight requires_grad= True
model.layers.16.mlp.Wo.weight requires_grad= True
model.layers.17.attn_norm.weight requires_grad= True
model.layers.17.attn.Wqkv.weight requires_grad= True
model.layers.17.attn.Wo.weight requires_grad= True
model.layers.17.mlp_norm.weight requires_grad= True
model.layers.17.mlp.Wi.weight requires_grad= True
model.layers.17.mlp.Wo.weight requires_grad= True
model.layers.18.attn_norm.weight requires_grad= True
model.layers.18.attn.Wqkv.weight requires_grad= True
model.layers.18.attn.Wo.weight requires_grad= True
model.layers.18.mlp_norm.weight requires_grad= True
model.layers.18.mlp.Wi.weight requires_grad= True
model.layers.18.mlp.Wo.weight requires_grad= True
model.layers.19.attn_norm.weight requires_grad= True
model.layers.19.attn.Wqkv.weight requires_grad= True
model.layers.19.attn.Wo.weight requires_grad= True
model.layers.19.mlp_norm.weight requires_grad= True
model.layers.19.mlp.Wi.weight requires_grad= True
model.layers.19.mlp.Wo.weight requires_grad= True
model.layers.20.attn_norm.weight requires_grad= True
model.layers.20.attn.Wqkv.weight requires_grad= True
model.layers.20.attn.Wo.weight requires_grad= True
model.layers.20.mlp_norm.weight requires_grad= True
model.layers.20.mlp.Wi.weight requires_grad= True
model.layers.20.mlp.Wo.weight requires_grad= True
model.layers.21.attn_norm.weight requires_grad= True
model.layers.21.attn.Wqkv.weight requires_grad= True
model.layers.21.attn.Wo.weight requires_grad= True
model.layers.21.mlp_norm.weight requires_grad= True
model.layers.21.mlp.Wi.weight requires_grad= True
model.layers.21.mlp.Wo.weight requires_grad= True
model.final_norm.weight requires_grad= True
head.dense.weight requires_grad= True
```

```
head.norm.weight requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True
```

[ ]:
```python
# Inspect the attention_mask tensor for the first few samples
for i in range(5):
    print(train_data_hf[i]['attention_mask'])
```

```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
```

[ ]:
```python
##########################################
layers_to_unfreeze = [
    "model.layers.21.attn_norm.weight",
    "model.layers.21.attn.Wqkv.weight",
    "model.layers.21.attn.Wo.weight",
    "model.layers.21.mlp_norm.weight",
    "model.layers.21.mlp.Wi.weight",
    "model.layers.21.mlp.Wo.weight",
    "model.final_norm.weight",
```

```
    "head.dense.weight",
    "head.norm.weight",
    "classifier.weight",
    "classifier.bias"]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("=============")
########################################
print("Experiment configuration used with this experiment:")
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
########################################
print("=============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```

```
ModernBertConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "ModernBertForMaskedLM"
  ],
  "attention_bias": false,
  "attention_dropout": 0.0,
  "bos_token_id": 50281,
  "classifier_activation": "gelu",
  "classifier_bias": false,
  "classifier_dropout": 0.0,
  "classifier_pooling": "mean",
  "cls_token_id": 50281,
  "decoder_bias": true,
  "deterministic_flash_attn": false,
  "embedding_dropout": 0.0,
  "eos_token_id": 50282,
  "global_attn_every_n_layers": 3,
  "global_rope_theta": 160000.0,
  "gradient_checkpointing": false,
  "hidden_activation": "gelu",
  "hidden_size": 768,
  "initializer_cutoff_factor": 2.0,
```

```
  "initializer_range": 0.02,
  "intermediate_size": 1152,
  "layer_norm_eps": 1e-05,
  "local_attention": 128,
  "local_rope_theta": 10000.0,
  "max_position_embeddings": 8192,
  "mlp_bias": false,
  "mlp_dropout": 0.0,
  "model_type": "modernbert",
  "norm_bias": false,
  "norm_eps": 1e-05,
  "num_attention_heads": 12,
  "num_hidden_layers": 22,
  "pad_token_id": 50283,
  "position_embedding_type": "absolute",
  "reference_compile": null,
  "repad_logits_with_grad": false,
  "sep_token_id": 50282,
  "sparse_pred_ignore_index": -100,
  "sparse_prediction": false,
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "vocab_size": 50368
}

=============
num_parameters: 149606402
num_trainable_parameters: 5607938
=============
Experiment configuration used with this experiment:
model used: answerdotai/ModernBERT-base
learning rate used: 5e-06
number of epochs: 1
maximum sequence length: 128
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
=============
num_trainable_parameters: 5607938
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
model.embeddings.tok_embeddings.weight requires_grad= False
model.embeddings.norm.weight requires_grad= False
model.layers.0.attn.Wqkv.weight requires_grad= False
```

```
model.layers.0.attn.Wo.weight requires_grad= False
model.layers.0.mlp_norm.weight requires_grad= False
model.layers.0.mlp.Wi.weight requires_grad= False
model.layers.0.mlp.Wo.weight requires_grad= False
model.layers.1.attn_norm.weight requires_grad= False
model.layers.1.attn.Wqkv.weight requires_grad= False
model.layers.1.attn.Wo.weight requires_grad= False
model.layers.1.mlp_norm.weight requires_grad= False
model.layers.1.mlp.Wi.weight requires_grad= False
model.layers.1.mlp.Wo.weight requires_grad= False
model.layers.2.attn_norm.weight requires_grad= False
model.layers.2.attn.Wqkv.weight requires_grad= False
model.layers.2.attn.Wo.weight requires_grad= False
model.layers.2.mlp_norm.weight requires_grad= False
model.layers.2.mlp.Wi.weight requires_grad= False
model.layers.2.mlp.Wo.weight requires_grad= False
model.layers.3.attn_norm.weight requires_grad= False
model.layers.3.attn.Wqkv.weight requires_grad= False
model.layers.3.attn.Wo.weight requires_grad= False
model.layers.3.mlp_norm.weight requires_grad= False
model.layers.3.mlp.Wi.weight requires_grad= False
model.layers.3.mlp.Wo.weight requires_grad= False
model.layers.4.attn_norm.weight requires_grad= False
model.layers.4.attn.Wqkv.weight requires_grad= False
model.layers.4.attn.Wo.weight requires_grad= False
model.layers.4.mlp_norm.weight requires_grad= False
model.layers.4.mlp.Wi.weight requires_grad= False
model.layers.4.mlp.Wo.weight requires_grad= False
model.layers.5.attn_norm.weight requires_grad= False
model.layers.5.attn.Wqkv.weight requires_grad= False
model.layers.5.attn.Wo.weight requires_grad= False
model.layers.5.mlp_norm.weight requires_grad= False
model.layers.5.mlp.Wi.weight requires_grad= False
model.layers.5.mlp.Wo.weight requires_grad= False
model.layers.6.attn_norm.weight requires_grad= False
model.layers.6.attn.Wqkv.weight requires_grad= False
model.layers.6.attn.Wo.weight requires_grad= False
model.layers.6.mlp_norm.weight requires_grad= False
model.layers.6.mlp.Wi.weight requires_grad= False
model.layers.6.mlp.Wo.weight requires_grad= False
model.layers.7.attn_norm.weight requires_grad= False
model.layers.7.attn.Wqkv.weight requires_grad= False
model.layers.7.attn.Wo.weight requires_grad= False
model.layers.7.mlp_norm.weight requires_grad= False
model.layers.7.mlp.Wi.weight requires_grad= False
model.layers.7.mlp.Wo.weight requires_grad= False
model.layers.8.attn_norm.weight requires_grad= False
model.layers.8.attn.Wqkv.weight requires_grad= False
```

```
model.layers.8.attn.Wo.weight requires_grad= False
model.layers.8.mlp_norm.weight requires_grad= False
model.layers.8.mlp.Wi.weight requires_grad= False
model.layers.8.mlp.Wo.weight requires_grad= False
model.layers.9.attn_norm.weight requires_grad= False
model.layers.9.attn.Wqkv.weight requires_grad= False
model.layers.9.attn.Wo.weight requires_grad= False
model.layers.9.mlp_norm.weight requires_grad= False
model.layers.9.mlp.Wi.weight requires_grad= False
model.layers.9.mlp.Wo.weight requires_grad= False
model.layers.10.attn_norm.weight requires_grad= False
model.layers.10.attn.Wqkv.weight requires_grad= False
model.layers.10.attn.Wo.weight requires_grad= False
model.layers.10.mlp_norm.weight requires_grad= False
model.layers.10.mlp.Wi.weight requires_grad= False
model.layers.10.mlp.Wo.weight requires_grad= False
model.layers.11.attn_norm.weight requires_grad= False
model.layers.11.attn.Wqkv.weight requires_grad= False
model.layers.11.attn.Wo.weight requires_grad= False
model.layers.11.mlp_norm.weight requires_grad= False
model.layers.11.mlp.Wi.weight requires_grad= False
model.layers.11.mlp.Wo.weight requires_grad= False
model.layers.12.attn_norm.weight requires_grad= False
model.layers.12.attn.Wqkv.weight requires_grad= False
model.layers.12.attn.Wo.weight requires_grad= False
model.layers.12.mlp_norm.weight requires_grad= False
model.layers.12.mlp.Wi.weight requires_grad= False
model.layers.12.mlp.Wo.weight requires_grad= False
model.layers.13.attn_norm.weight requires_grad= False
model.layers.13.attn.Wqkv.weight requires_grad= False
model.layers.13.attn.Wo.weight requires_grad= False
model.layers.13.mlp_norm.weight requires_grad= False
model.layers.13.mlp.Wi.weight requires_grad= False
model.layers.13.mlp.Wo.weight requires_grad= False
model.layers.14.attn_norm.weight requires_grad= False
model.layers.14.attn.Wqkv.weight requires_grad= False
model.layers.14.attn.Wo.weight requires_grad= False
model.layers.14.mlp_norm.weight requires_grad= False
model.layers.14.mlp.Wi.weight requires_grad= False
model.layers.14.mlp.Wo.weight requires_grad= False
model.layers.15.attn_norm.weight requires_grad= False
model.layers.15.attn.Wqkv.weight requires_grad= False
model.layers.15.attn.Wo.weight requires_grad= False
model.layers.15.mlp_norm.weight requires_grad= False
model.layers.15.mlp.Wi.weight requires_grad= False
model.layers.15.mlp.Wo.weight requires_grad= False
model.layers.16.attn_norm.weight requires_grad= False
model.layers.16.attn.Wqkv.weight requires_grad= False
```

```
model.layers.16.attn.Wo.weight requires_grad= False
model.layers.16.mlp_norm.weight requires_grad= False
model.layers.16.mlp.Wi.weight requires_grad= False
model.layers.16.mlp.Wo.weight requires_grad= False
model.layers.17.attn_norm.weight requires_grad= False
model.layers.17.attn.Wqkv.weight requires_grad= False
model.layers.17.attn.Wo.weight requires_grad= False
model.layers.17.mlp_norm.weight requires_grad= False
model.layers.17.mlp.Wi.weight requires_grad= False
model.layers.17.mlp.Wo.weight requires_grad= False
model.layers.18.attn_norm.weight requires_grad= False
model.layers.18.attn.Wqkv.weight requires_grad= False
model.layers.18.attn.Wo.weight requires_grad= False
model.layers.18.mlp_norm.weight requires_grad= False
model.layers.18.mlp.Wi.weight requires_grad= False
model.layers.18.mlp.Wo.weight requires_grad= False
model.layers.19.attn_norm.weight requires_grad= False
model.layers.19.attn.Wqkv.weight requires_grad= False
model.layers.19.attn.Wo.weight requires_grad= False
model.layers.19.mlp_norm.weight requires_grad= False
model.layers.19.mlp.Wi.weight requires_grad= False
model.layers.19.mlp.Wo.weight requires_grad= False
model.layers.20.attn_norm.weight requires_grad= False
model.layers.20.attn.Wqkv.weight requires_grad= False
model.layers.20.attn.Wo.weight requires_grad= False
model.layers.20.mlp_norm.weight requires_grad= False
model.layers.20.mlp.Wi.weight requires_grad= False
model.layers.20.mlp.Wo.weight requires_grad= False
model.layers.21.attn_norm.weight requires_grad= True
model.layers.21.attn.Wqkv.weight requires_grad= True
model.layers.21.attn.Wo.weight requires_grad= True
model.layers.21.mlp_norm.weight requires_grad= True
model.layers.21.mlp.Wi.weight requires_grad= True
model.layers.21.mlp.Wo.weight requires_grad= True
model.final_norm.weight requires_grad= True
head.dense.weight requires_grad= True
head.norm.weight requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True
```

[ ]:
```
# model.resize_token_embeddings(len(tokenizer))
```

[ ]:
```
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
```

```
    val_dataset = val_data_hf,
    output_dir = dir_results,
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-20-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
/usr/local/lib/python3.11/dist-packages/torch/_inductor/compile_fx.py:194:
UserWarning: TensorFloat32 tensor cores for float32 matrix multiplication
available but not enabled. Consider setting
`torch.set_float32_matmul_precision('high')` for better performance.
  warnings.warn(
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 1.299619436264038, 'eval_accuracy':
0.4513064133016627, 'eval_precision': 0.4474393530997305, 'eval_recall':
0.8645833333333334, 'eval_f1': 0.5896980461811723, 'eval_runtime': 1.9172,
'eval_samples_per_second': 219.59, 'eval_steps_per_second': 2.086, 'epoch': 1.0}
Test metrics: {'eval_loss': 1.2373243570327759, 'eval_accuracy':
0.49073064340239914, 'eval_precision': 0.4839901477832512, 'eval_recall':
0.891156462585034, 'eval_f1': 0.627294493216281, 'eval_runtime': 2.9097,
'eval_samples_per_second': 315.153, 'eval_steps_per_second': 2.749, 'epoch':
1.0}
```

[ ]:
```
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
  ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
```

```
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to:
/content/drive/MyDrive/266-final/models/single_answerdotai/ModernBERT-
base_binary_complexity_20250411_121436

<IPython.core.display.HTML object>

EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```

0.2.8 snc answerdotai/ModernBERT-large regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[ ]:
```
# Define Experiment Parameters
# named_model = "bert-base-cased"
# named_model = "roberta-base"
# named_model = "bert-large-cased"
# named_model = "roberta-large"
named_model = "answerdotai/ModernBERT-large" # modern bert
#############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
num_epochs = 1
#########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
```

```
# x_col = "dep_sequence"
# x_col = "morph_sequence"
############
y_col = "binary_complexity"
# y_col = "complexity"
############
x_task = "single"
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="answerdotai/ModernBERT-large",
    local_model_path=None,
    config=None)
############
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
```

```
#     local_model_path="...CONFIGURE_PATH...",
#     config=custom_config)
##########################################
print("=============")
print(named_model, ":")
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters at load:", model.
  ↪num_parameters(only_trainable=True))
print("=============")
print("model lineage:", MODEL_LINEAGE)
print("=============")
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]

Map:   0%|          | 0/421 [00:00<?, ? examples/s]

Map:   0%|          | 0/917 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([50281,  1989,   281,   253, 15196,
273, 29732,   506,   344,  3534,
         5293,    13,   984,   253,  2579,   273,   253, 35514, 20651,   281,
          731,    28,   597,  4824,   352,   327,   616, 13574,    15, 50282,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283]),
'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])}
Loading from Hugging Face model: answerdotai/ModernBERT-large

tokenizer_config.json:   0%|          | 0.00/20.8k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/2.13M [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/694 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/1.19k [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/1.58G [00:00<?, ?B/s]
```

```
Some weights of ModernBertForSequenceClassification were not initialized from
the model checkpoint at answerdotai/ModernBERT-large and are newly initialized:
['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

=============
answerdotai/ModernBERT-large :
=============
num_parameters: 395833346
num_trainable_parameters at load: 395833346
=============
model lineage: {'type': 'huggingface_hub', 'path': 'answerdotai/ModernBERT-
large', 'timestamp': '2025-04-11 12:15:14'}
=============
```

[ ]:
```
print(model)
```

```
ModernBertForSequenceClassification(
  (model): ModernBertModel(
    (embeddings): ModernBertEmbeddings(
      (tok_embeddings): Embedding(50368, 1024, padding_idx=50283)
      (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
      (drop): Dropout(p=0.0, inplace=False)
    )
    (layers): ModuleList(
      (0): ModernBertEncoderLayer(
        (attn_norm): Identity()
        (attn): ModernBertAttention(
          (Wqkv): Linear(in_features=1024, out_features=3072, bias=False)
          (rotary_emb): ModernBertRotaryEmbedding()
          (Wo): Linear(in_features=1024, out_features=1024, bias=False)
          (out_drop): Identity()
        )
        (mlp_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        (mlp): ModernBertMLP(
          (Wi): Linear(in_features=1024, out_features=5248, bias=False)
          (act): GELUActivation()
          (drop): Dropout(p=0.0, inplace=False)
          (Wo): Linear(in_features=2624, out_features=1024, bias=False)
        )
      )
      (1-27): 27 x ModernBertEncoderLayer(
        (attn_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        (attn): ModernBertAttention(
          (Wqkv): Linear(in_features=1024, out_features=3072, bias=False)
          (rotary_emb): ModernBertRotaryEmbedding()
          (Wo): Linear(in_features=1024, out_features=1024, bias=False)
          (out_drop): Identity()
```

```
        )
        (mlp_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        (mlp): ModernBertMLP(
          (Wi): Linear(in_features=1024, out_features=5248, bias=False)
          (act): GELUActivation()
          (drop): Dropout(p=0.0, inplace=False)
          (Wo): Linear(in_features=2624, out_features=1024, bias=False)
        )
      )
    )
    (final_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (head): ModernBertPredictionHead(
    (dense): Linear(in_features=1024, out_features=1024, bias=False)
    (act): GELUActivation()
    (norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  )
  (drop): Dropout(p=0.0, inplace=False)
  (classifier): Linear(in_features=1024, out_features=2, bias=True)
)
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
model.embeddings.tok_embeddings.weight requires_grad= True
model.embeddings.norm.weight requires_grad= True
model.layers.0.attn.Wqkv.weight requires_grad= True
model.layers.0.attn.Wo.weight requires_grad= True
model.layers.0.mlp_norm.weight requires_grad= True
model.layers.0.mlp.Wi.weight requires_grad= True
model.layers.0.mlp.Wo.weight requires_grad= True
model.layers.1.attn_norm.weight requires_grad= True
model.layers.1.attn.Wqkv.weight requires_grad= True
model.layers.1.attn.Wo.weight requires_grad= True
model.layers.1.mlp_norm.weight requires_grad= True
model.layers.1.mlp.Wi.weight requires_grad= True
model.layers.1.mlp.Wo.weight requires_grad= True
model.layers.2.attn_norm.weight requires_grad= True
model.layers.2.attn.Wqkv.weight requires_grad= True
model.layers.2.attn.Wo.weight requires_grad= True
model.layers.2.mlp_norm.weight requires_grad= True
model.layers.2.mlp.Wi.weight requires_grad= True
model.layers.2.mlp.Wo.weight requires_grad= True
model.layers.3.attn_norm.weight requires_grad= True
model.layers.3.attn.Wqkv.weight requires_grad= True
model.layers.3.attn.Wo.weight requires_grad= True
model.layers.3.mlp_norm.weight requires_grad= True
model.layers.3.mlp.Wi.weight requires_grad= True
```

```
model.layers.3.mlp.Wo.weight requires_grad= True
model.layers.4.attn_norm.weight requires_grad= True
model.layers.4.attn.Wqkv.weight requires_grad= True
model.layers.4.attn.Wo.weight requires_grad= True
model.layers.4.mlp_norm.weight requires_grad= True
model.layers.4.mlp.Wi.weight requires_grad= True
model.layers.4.mlp.Wo.weight requires_grad= True
model.layers.5.attn_norm.weight requires_grad= True
model.layers.5.attn.Wqkv.weight requires_grad= True
model.layers.5.attn.Wo.weight requires_grad= True
model.layers.5.mlp_norm.weight requires_grad= True
model.layers.5.mlp.Wi.weight requires_grad= True
model.layers.5.mlp.Wo.weight requires_grad= True
model.layers.6.attn_norm.weight requires_grad= True
model.layers.6.attn.Wqkv.weight requires_grad= True
model.layers.6.attn.Wo.weight requires_grad= True
model.layers.6.mlp_norm.weight requires_grad= True
model.layers.6.mlp.Wi.weight requires_grad= True
model.layers.6.mlp.Wo.weight requires_grad= True
model.layers.7.attn_norm.weight requires_grad= True
model.layers.7.attn.Wqkv.weight requires_grad= True
model.layers.7.attn.Wo.weight requires_grad= True
model.layers.7.mlp_norm.weight requires_grad= True
model.layers.7.mlp.Wi.weight requires_grad= True
model.layers.7.mlp.Wo.weight requires_grad= True
model.layers.8.attn_norm.weight requires_grad= True
model.layers.8.attn.Wqkv.weight requires_grad= True
model.layers.8.attn.Wo.weight requires_grad= True
model.layers.8.mlp_norm.weight requires_grad= True
model.layers.8.mlp.Wi.weight requires_grad= True
model.layers.8.mlp.Wo.weight requires_grad= True
model.layers.9.attn_norm.weight requires_grad= True
model.layers.9.attn.Wqkv.weight requires_grad= True
model.layers.9.attn.Wo.weight requires_grad= True
model.layers.9.mlp_norm.weight requires_grad= True
model.layers.9.mlp.Wi.weight requires_grad= True
model.layers.9.mlp.Wo.weight requires_grad= True
model.layers.10.attn_norm.weight requires_grad= True
model.layers.10.attn.Wqkv.weight requires_grad= True
model.layers.10.attn.Wo.weight requires_grad= True
model.layers.10.mlp_norm.weight requires_grad= True
model.layers.10.mlp.Wi.weight requires_grad= True
model.layers.10.mlp.Wo.weight requires_grad= True
model.layers.11.attn_norm.weight requires_grad= True
model.layers.11.attn.Wqkv.weight requires_grad= True
model.layers.11.attn.Wo.weight requires_grad= True
model.layers.11.mlp_norm.weight requires_grad= True
model.layers.11.mlp.Wi.weight requires_grad= True
```

```
model.layers.11.mlp.Wo.weight requires_grad= True
model.layers.12.attn_norm.weight requires_grad= True
model.layers.12.attn.Wqkv.weight requires_grad= True
model.layers.12.attn.Wo.weight requires_grad= True
model.layers.12.mlp_norm.weight requires_grad= True
model.layers.12.mlp.Wi.weight requires_grad= True
model.layers.12.mlp.Wo.weight requires_grad= True
model.layers.13.attn_norm.weight requires_grad= True
model.layers.13.attn.Wqkv.weight requires_grad= True
model.layers.13.attn.Wo.weight requires_grad= True
model.layers.13.mlp_norm.weight requires_grad= True
model.layers.13.mlp.Wi.weight requires_grad= True
model.layers.13.mlp.Wo.weight requires_grad= True
model.layers.14.attn_norm.weight requires_grad= True
model.layers.14.attn.Wqkv.weight requires_grad= True
model.layers.14.attn.Wo.weight requires_grad= True
model.layers.14.mlp_norm.weight requires_grad= True
model.layers.14.mlp.Wi.weight requires_grad= True
model.layers.14.mlp.Wo.weight requires_grad= True
model.layers.15.attn_norm.weight requires_grad= True
model.layers.15.attn.Wqkv.weight requires_grad= True
model.layers.15.attn.Wo.weight requires_grad= True
model.layers.15.mlp_norm.weight requires_grad= True
model.layers.15.mlp.Wi.weight requires_grad= True
model.layers.15.mlp.Wo.weight requires_grad= True
model.layers.16.attn_norm.weight requires_grad= True
model.layers.16.attn.Wqkv.weight requires_grad= True
model.layers.16.attn.Wo.weight requires_grad= True
model.layers.16.mlp_norm.weight requires_grad= True
model.layers.16.mlp.Wi.weight requires_grad= True
model.layers.16.mlp.Wo.weight requires_grad= True
model.layers.17.attn_norm.weight requires_grad= True
model.layers.17.attn.Wqkv.weight requires_grad= True
model.layers.17.attn.Wo.weight requires_grad= True
model.layers.17.mlp_norm.weight requires_grad= True
model.layers.17.mlp.Wi.weight requires_grad= True
model.layers.17.mlp.Wo.weight requires_grad= True
model.layers.18.attn_norm.weight requires_grad= True
model.layers.18.attn.Wqkv.weight requires_grad= True
model.layers.18.attn.Wo.weight requires_grad= True
model.layers.18.mlp_norm.weight requires_grad= True
model.layers.18.mlp.Wi.weight requires_grad= True
model.layers.18.mlp.Wo.weight requires_grad= True
model.layers.19.attn_norm.weight requires_grad= True
model.layers.19.attn.Wqkv.weight requires_grad= True
model.layers.19.attn.Wo.weight requires_grad= True
model.layers.19.mlp_norm.weight requires_grad= True
model.layers.19.mlp.Wi.weight requires_grad= True
```

```
model.layers.19.mlp.Wo.weight requires_grad= True
model.layers.20.attn_norm.weight requires_grad= True
model.layers.20.attn.Wqkv.weight requires_grad= True
model.layers.20.attn.Wo.weight requires_grad= True
model.layers.20.mlp_norm.weight requires_grad= True
model.layers.20.mlp.Wi.weight requires_grad= True
model.layers.20.mlp.Wo.weight requires_grad= True
model.layers.21.attn_norm.weight requires_grad= True
model.layers.21.attn.Wqkv.weight requires_grad= True
model.layers.21.attn.Wo.weight requires_grad= True
model.layers.21.mlp_norm.weight requires_grad= True
model.layers.21.mlp.Wi.weight requires_grad= True
model.layers.21.mlp.Wo.weight requires_grad= True
model.layers.22.attn_norm.weight requires_grad= True
model.layers.22.attn.Wqkv.weight requires_grad= True
model.layers.22.attn.Wo.weight requires_grad= True
model.layers.22.mlp_norm.weight requires_grad= True
model.layers.22.mlp.Wi.weight requires_grad= True
model.layers.22.mlp.Wo.weight requires_grad= True
model.layers.23.attn_norm.weight requires_grad= True
model.layers.23.attn.Wqkv.weight requires_grad= True
model.layers.23.attn.Wo.weight requires_grad= True
model.layers.23.mlp_norm.weight requires_grad= True
model.layers.23.mlp.Wi.weight requires_grad= True
model.layers.23.mlp.Wo.weight requires_grad= True
model.layers.24.attn_norm.weight requires_grad= True
model.layers.24.attn.Wqkv.weight requires_grad= True
model.layers.24.attn.Wo.weight requires_grad= True
model.layers.24.mlp_norm.weight requires_grad= True
model.layers.24.mlp.Wi.weight requires_grad= True
model.layers.24.mlp.Wo.weight requires_grad= True
model.layers.25.attn_norm.weight requires_grad= True
model.layers.25.attn.Wqkv.weight requires_grad= True
model.layers.25.attn.Wo.weight requires_grad= True
model.layers.25.mlp_norm.weight requires_grad= True
model.layers.25.mlp.Wi.weight requires_grad= True
model.layers.25.mlp.Wo.weight requires_grad= True
model.layers.26.attn_norm.weight requires_grad= True
model.layers.26.attn.Wqkv.weight requires_grad= True
model.layers.26.attn.Wo.weight requires_grad= True
model.layers.26.mlp_norm.weight requires_grad= True
model.layers.26.mlp.Wi.weight requires_grad= True
model.layers.26.mlp.Wo.weight requires_grad= True
model.layers.27.attn_norm.weight requires_grad= True
model.layers.27.attn.Wqkv.weight requires_grad= True
model.layers.27.attn.Wo.weight requires_grad= True
model.layers.27.mlp_norm.weight requires_grad= True
model.layers.27.mlp.Wi.weight requires_grad= True
```

```
model.layers.27.mlp.Wo.weight requires_grad= True
model.final_norm.weight requires_grad= True
head.dense.weight requires_grad= True
head.norm.weight requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True
```

[ ]:
```python
# Inspect the attention_mask tensor for the first few samples
for i in range(5):
    print(train_data_hf[i]['attention_mask'])
```

```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])
```

[ ]:
```python
##########################################
layers_to_unfreeze = [
    "model.layers.27.attn_norm.weight",
    "model.layers.27.attn.Wqkv.weight",
    "model.layers.27.attn.Wo.weight",
    "model.layers.27.mlp_norm.weight",
```

```
    "model.layers.27.mlp.Wi.weight",
    "model.layers.27.mlp.Wo.weight",
    "model.final_norm.weight",
    "head.dense.weight",
    "head.norm.weight",
    "classifier.weight",
    "classifier.bias"
    ]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("=============")
########################################
print("Experiment configuration used with this experiment:")
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
########################################
print("=============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```

```
ModernBertConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "ModernBertForMaskedLM"
  ],
  "attention_bias": false,
  "attention_dropout": 0.0,
  "bos_token_id": 50281,
  "classifier_activation": "gelu",
  "classifier_bias": false,
  "classifier_dropout": 0.0,
  "classifier_pooling": "mean",
  "cls_token_id": 50281,
  "decoder_bias": true,
  "deterministic_flash_attn": false,
  "embedding_dropout": 0.0,
  "eos_token_id": 50282,
  "global_attn_every_n_layers": 3,
  "global_rope_theta": 160000.0,
```

```
  "gradient_checkpointing": false,
  "hidden_activation": "gelu",
  "hidden_size": 1024,
  "initializer_cutoff_factor": 2.0,
  "initializer_range": 0.02,
  "intermediate_size": 2624,
  "layer_norm_eps": 1e-05,
  "local_attention": 128,
  "local_rope_theta": 10000.0,
  "max_position_embeddings": 8192,
  "mlp_bias": false,
  "mlp_dropout": 0.0,
  "model_type": "modernbert",
  "norm_bias": false,
  "norm_eps": 1e-05,
  "num_attention_heads": 16,
  "num_hidden_layers": 28,
  "pad_token_id": 50283,
  "position_embedding_type": "absolute",
  "reference_compile": null,
  "repad_logits_with_grad": false,
  "sep_token_id": 50282,
  "sparse_pred_ignore_index": -100,
  "sparse_prediction": false,
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "vocab_size": 50368
}

=============
num_parameters: 395833346
num_trainable_parameters: 13309954
=============
Experiment configuration used with this experiment:
model used: answerdotai/ModernBERT-large
learning rate used: 5e-06
number of epochs: 1
maximum sequence length: 128
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
=============
num_trainable_parameters: 13309954
```

```
[ ]: for name, param in model.named_parameters():
         print(name, "requires_grad=", param.requires_grad)
```

```
model.embeddings.tok_embeddings.weight requires_grad= False
model.embeddings.norm.weight requires_grad= False
model.layers.0.attn.Wqkv.weight requires_grad= False
model.layers.0.attn.Wo.weight requires_grad= False
model.layers.0.mlp_norm.weight requires_grad= False
model.layers.0.mlp.Wi.weight requires_grad= False
model.layers.0.mlp.Wo.weight requires_grad= False
model.layers.1.attn_norm.weight requires_grad= False
model.layers.1.attn.Wqkv.weight requires_grad= False
model.layers.1.attn.Wo.weight requires_grad= False
model.layers.1.mlp_norm.weight requires_grad= False
model.layers.1.mlp.Wi.weight requires_grad= False
model.layers.1.mlp.Wo.weight requires_grad= False
model.layers.2.attn_norm.weight requires_grad= False
model.layers.2.attn.Wqkv.weight requires_grad= False
model.layers.2.attn.Wo.weight requires_grad= False
model.layers.2.mlp_norm.weight requires_grad= False
model.layers.2.mlp.Wi.weight requires_grad= False
model.layers.2.mlp.Wo.weight requires_grad= False
model.layers.3.attn_norm.weight requires_grad= False
model.layers.3.attn.Wqkv.weight requires_grad= False
model.layers.3.attn.Wo.weight requires_grad= False
model.layers.3.mlp_norm.weight requires_grad= False
model.layers.3.mlp.Wi.weight requires_grad= False
model.layers.3.mlp.Wo.weight requires_grad= False
model.layers.4.attn_norm.weight requires_grad= False
model.layers.4.attn.Wqkv.weight requires_grad= False
model.layers.4.attn.Wo.weight requires_grad= False
model.layers.4.mlp_norm.weight requires_grad= False
model.layers.4.mlp.Wi.weight requires_grad= False
model.layers.4.mlp.Wo.weight requires_grad= False
model.layers.5.attn_norm.weight requires_grad= False
model.layers.5.attn.Wqkv.weight requires_grad= False
model.layers.5.attn.Wo.weight requires_grad= False
model.layers.5.mlp_norm.weight requires_grad= False
model.layers.5.mlp.Wi.weight requires_grad= False
model.layers.5.mlp.Wo.weight requires_grad= False
model.layers.6.attn_norm.weight requires_grad= False
model.layers.6.attn.Wqkv.weight requires_grad= False
model.layers.6.attn.Wo.weight requires_grad= False
model.layers.6.mlp_norm.weight requires_grad= False
model.layers.6.mlp.Wi.weight requires_grad= False
model.layers.6.mlp.Wo.weight requires_grad= False
model.layers.7.attn_norm.weight requires_grad= False
model.layers.7.attn.Wqkv.weight requires_grad= False
```

```
model.layers.7.attn.Wo.weight requires_grad= False
model.layers.7.mlp_norm.weight requires_grad= False
model.layers.7.mlp.Wi.weight requires_grad= False
model.layers.7.mlp.Wo.weight requires_grad= False
model.layers.8.attn_norm.weight requires_grad= False
model.layers.8.attn.Wqkv.weight requires_grad= False
model.layers.8.attn.Wo.weight requires_grad= False
model.layers.8.mlp_norm.weight requires_grad= False
model.layers.8.mlp.Wi.weight requires_grad= False
model.layers.8.mlp.Wo.weight requires_grad= False
model.layers.9.attn_norm.weight requires_grad= False
model.layers.9.attn.Wqkv.weight requires_grad= False
model.layers.9.attn.Wo.weight requires_grad= False
model.layers.9.mlp_norm.weight requires_grad= False
model.layers.9.mlp.Wi.weight requires_grad= False
model.layers.9.mlp.Wo.weight requires_grad= False
model.layers.10.attn_norm.weight requires_grad= False
model.layers.10.attn.Wqkv.weight requires_grad= False
model.layers.10.attn.Wo.weight requires_grad= False
model.layers.10.mlp_norm.weight requires_grad= False
model.layers.10.mlp.Wi.weight requires_grad= False
model.layers.10.mlp.Wo.weight requires_grad= False
model.layers.11.attn_norm.weight requires_grad= False
model.layers.11.attn.Wqkv.weight requires_grad= False
model.layers.11.attn.Wo.weight requires_grad= False
model.layers.11.mlp_norm.weight requires_grad= False
model.layers.11.mlp.Wi.weight requires_grad= False
model.layers.11.mlp.Wo.weight requires_grad= False
model.layers.12.attn_norm.weight requires_grad= False
model.layers.12.attn.Wqkv.weight requires_grad= False
model.layers.12.attn.Wo.weight requires_grad= False
model.layers.12.mlp_norm.weight requires_grad= False
model.layers.12.mlp.Wi.weight requires_grad= False
model.layers.12.mlp.Wo.weight requires_grad= False
model.layers.13.attn_norm.weight requires_grad= False
model.layers.13.attn.Wqkv.weight requires_grad= False
model.layers.13.attn.Wo.weight requires_grad= False
model.layers.13.mlp_norm.weight requires_grad= False
model.layers.13.mlp.Wi.weight requires_grad= False
model.layers.13.mlp.Wo.weight requires_grad= False
model.layers.14.attn_norm.weight requires_grad= False
model.layers.14.attn.Wqkv.weight requires_grad= False
model.layers.14.attn.Wo.weight requires_grad= False
model.layers.14.mlp_norm.weight requires_grad= False
model.layers.14.mlp.Wi.weight requires_grad= False
model.layers.14.mlp.Wo.weight requires_grad= False
model.layers.15.attn_norm.weight requires_grad= False
model.layers.15.attn.Wqkv.weight requires_grad= False
```

```
model.layers.15.attn.Wo.weight requires_grad= False
model.layers.15.mlp_norm.weight requires_grad= False
model.layers.15.mlp.Wi.weight requires_grad= False
model.layers.15.mlp.Wo.weight requires_grad= False
model.layers.16.attn_norm.weight requires_grad= False
model.layers.16.attn.Wqkv.weight requires_grad= False
model.layers.16.attn.Wo.weight requires_grad= False
model.layers.16.mlp_norm.weight requires_grad= False
model.layers.16.mlp.Wi.weight requires_grad= False
model.layers.16.mlp.Wo.weight requires_grad= False
model.layers.17.attn_norm.weight requires_grad= False
model.layers.17.attn.Wqkv.weight requires_grad= False
model.layers.17.attn.Wo.weight requires_grad= False
model.layers.17.mlp_norm.weight requires_grad= False
model.layers.17.mlp.Wi.weight requires_grad= False
model.layers.17.mlp.Wo.weight requires_grad= False
model.layers.18.attn_norm.weight requires_grad= False
model.layers.18.attn.Wqkv.weight requires_grad= False
model.layers.18.attn.Wo.weight requires_grad= False
model.layers.18.mlp_norm.weight requires_grad= False
model.layers.18.mlp.Wi.weight requires_grad= False
model.layers.18.mlp.Wo.weight requires_grad= False
model.layers.19.attn_norm.weight requires_grad= False
model.layers.19.attn.Wqkv.weight requires_grad= False
model.layers.19.attn.Wo.weight requires_grad= False
model.layers.19.mlp_norm.weight requires_grad= False
model.layers.19.mlp.Wi.weight requires_grad= False
model.layers.19.mlp.Wo.weight requires_grad= False
model.layers.20.attn_norm.weight requires_grad= False
model.layers.20.attn.Wqkv.weight requires_grad= False
model.layers.20.attn.Wo.weight requires_grad= False
model.layers.20.mlp_norm.weight requires_grad= False
model.layers.20.mlp.Wi.weight requires_grad= False
model.layers.20.mlp.Wo.weight requires_grad= False
model.layers.21.attn_norm.weight requires_grad= False
model.layers.21.attn.Wqkv.weight requires_grad= False
model.layers.21.attn.Wo.weight requires_grad= False
model.layers.21.mlp_norm.weight requires_grad= False
model.layers.21.mlp.Wi.weight requires_grad= False
model.layers.21.mlp.Wo.weight requires_grad= False
model.layers.22.attn_norm.weight requires_grad= False
model.layers.22.attn.Wqkv.weight requires_grad= False
model.layers.22.attn.Wo.weight requires_grad= False
model.layers.22.mlp_norm.weight requires_grad= False
model.layers.22.mlp.Wi.weight requires_grad= False
model.layers.22.mlp.Wo.weight requires_grad= False
model.layers.23.attn_norm.weight requires_grad= False
model.layers.23.attn.Wqkv.weight requires_grad= False
```

```
model.layers.23.attn.Wo.weight requires_grad= False
model.layers.23.mlp_norm.weight requires_grad= False
model.layers.23.mlp.Wi.weight requires_grad= False
model.layers.23.mlp.Wo.weight requires_grad= False
model.layers.24.attn_norm.weight requires_grad= False
model.layers.24.attn.Wqkv.weight requires_grad= False
model.layers.24.attn.Wo.weight requires_grad= False
model.layers.24.mlp_norm.weight requires_grad= False
model.layers.24.mlp.Wi.weight requires_grad= False
model.layers.24.mlp.Wo.weight requires_grad= False
model.layers.25.attn_norm.weight requires_grad= False
model.layers.25.attn.Wqkv.weight requires_grad= False
model.layers.25.attn.Wo.weight requires_grad= False
model.layers.25.mlp_norm.weight requires_grad= False
model.layers.25.mlp.Wi.weight requires_grad= False
model.layers.25.mlp.Wo.weight requires_grad= False
model.layers.26.attn_norm.weight requires_grad= False
model.layers.26.attn.Wqkv.weight requires_grad= False
model.layers.26.attn.Wo.weight requires_grad= False
model.layers.26.mlp_norm.weight requires_grad= False
model.layers.26.mlp.Wi.weight requires_grad= False
model.layers.26.mlp.Wo.weight requires_grad= False
model.layers.27.attn_norm.weight requires_grad= True
model.layers.27.attn.Wqkv.weight requires_grad= True
model.layers.27.attn.Wo.weight requires_grad= True
model.layers.27.mlp_norm.weight requires_grad= True
model.layers.27.mlp.Wi.weight requires_grad= True
model.layers.27.mlp.Wo.weight requires_grad= True
model.final_norm.weight requires_grad= True
head.dense.weight requires_grad= True
head.norm.weight requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True
```

[ ]:
```
# model.resize_token_embeddings(len(tokenizer))
```

[ ]:
```
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
    val_dataset = val_data_hf,
    output_dir = dir_results,
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
```

```
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-20-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

```
<IPython.core.display.HTML object>
```

```
W0411 12:16:04.688000 6494 torch/_dynamo/convert_frame.py:906] [1/8]
torch._dynamo hit config.cache_size_limit (8)
W0411 12:16:04.688000 6494 torch/_dynamo/convert_frame.py:906] [1/8]
function: 'compiled_mlp' (/usr/local/lib/python3.11/dist-
packages/transformers/models/modernbert/modeling_modernbert.py:552)
W0411 12:16:04.688000 6494 torch/_dynamo/convert_frame.py:906] [1/8]    last
reason: 1/0: GLOBAL_STATE changed: grad_mode
W0411 12:16:04.688000 6494 torch/_dynamo/convert_frame.py:906] [1/8] To log all
recompilation reasons, use TORCH_LOGS="recompiles".
W0411 12:16:04.688000 6494 torch/_dynamo/convert_frame.py:906] [1/8] To diagnose
recompilation issues, see
https://pytorch.org/docs/main/torch.compiler_troubleshooting.html.
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 0.7014551758766174, 'eval_accuracy':
0.5463182897862233, 'eval_precision': 0.5024390243902439, 'eval_recall':
0.5364583333333334, 'eval_f1': 0.5188916876574308, 'eval_runtime': 3.3411,
'eval_samples_per_second': 126.005, 'eval_steps_per_second': 1.197, 'epoch':
1.0}
Test metrics: {'eval_loss': 0.7242145538330078, 'eval_accuracy':
0.4940021810250818, 'eval_precision': 0.4747252747252747, 'eval_recall':
0.4897959183673469, 'eval_f1': 0.48214285714285715, 'eval_runtime': 6.0275,
'eval_samples_per_second': 152.136, 'eval_steps_per_second': 1.327, 'epoch':
1.0}
```

[ ]:
```
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
  ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
```

```
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to:
/content/drive/MyDrive/266-final/models/single_answerdotai/ModernBERT-
large_binary_complexity_20250411_121617

<IPython.core.display.HTML object>

EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```

0.2.9 snc microsoft/deberta-v3-base regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[ ]:
```
# Define Experiment Parameters
# named_model = "bert-base-cased"
# named_model = "roberta-base"
# named_model = "bert-large-cased"
# named_model = "roberta-large"
# named_model = "answerdotai/ModernBERT-base" # modern bert
# named_model = "answerdotai/ModernBERT-large" # modern bert
named_model = "microsoft/deberta-v3-base" # deberta
############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
```

```
num_epochs = 1
##########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
############
y_col = "binary_complexity"
# y_col = "complexity"
############
x_task = "single"
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
##########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
##########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="microsoft/deberta-v3-base",
```

```
    local_model_path=None,
    config=None)
############
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
#     local_model_path="...CONFIGURE_PATH...",
#     config=custom_config)
###########################################
print("=============")
print(named_model, ":")
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters at load:", model.
 ↪num_parameters(only_trainable=True))
print("=============")
print("model lineage:", MODEL_LINEAGE)
print("=============")
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]

Map:   0%|          | 0/421 [00:00<?, ? examples/s]

Map:   0%|          | 0/917 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([50281,  1989,   281,   253, 15196,
273, 29732,   506,   344,  3534,
         5293,    13,   984,   253,  2579,   273,   253, 35514, 20651,   281,
          731,    28,   597,  4824,   352,   327,   616, 13574,    15, 50282,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283,
        50283, 50283, 50283, 50283, 50283, 50283, 50283, 50283]),
'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])}
Loading from Hugging Face model: microsoft/deberta-v3-base

tokenizer_config.json:   0%|          | 0.00/52.0 [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/579 [00:00<?, ?B/s]
```

```
spm.model:   0%|          | 0.00/2.46M [00:00<?, ?B/s]
```

```
/usr/local/lib/python3.11/dist-
packages/transformers/convert_slow_tokenizer.py:559: UserWarning: The
sentencepiece tokenizer that you are converting to a fast tokenizer uses the
byte fallback option which is not implemented in the fast tokenizers. In
practice this means that the fast version of the tokenizer can produce unknown
tokens whereas the sentencepiece version would have converted these unknown
tokens into a sequence of byte tokens matching the original piece of text.
  warnings.warn(
```

```
pytorch_model.bin:   0%|          | 0.00/371M [00:00<?, ?B/s]
```

```
Some weights of DebertaV2ForSequenceClassification were not initialized from the
model checkpoint at microsoft/deberta-v3-base and are newly initialized:
['classifier.bias', 'classifier.weight', 'pooler.dense.bias',
'pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
=============
microsoft/deberta-v3-base :
=============
num_parameters: 184423682
num_trainable_parameters at load: 184423682
=============
model lineage: {'type': 'huggingface_hub', 'path': 'microsoft/deberta-v3-base',
'timestamp': '2025-04-11 12:17:22'}
=============
```

[ ]:
```
print(model)
```

```
DebertaV2ForSequenceClassification(
  (deberta): DebertaV2Model(
    (embeddings): DebertaV2Embeddings(
      (word_embeddings): Embedding(128100, 768, padding_idx=0)
      (LayerNorm): LayerNorm((768,), eps=1e-07, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): DebertaV2Encoder(
      (layer): ModuleList(
        (0-11): 12 x DebertaV2Layer(
          (attention): DebertaV2Attention(
            (self): DisentangledSelfAttention(
              (query_proj): Linear(in_features=768, out_features=768, bias=True)
              (key_proj): Linear(in_features=768, out_features=768, bias=True)
              (value_proj): Linear(in_features=768, out_features=768, bias=True)
              (pos_dropout): Dropout(p=0.1, inplace=False)
              (dropout): Dropout(p=0.1, inplace=False)
```

```
            )
            (output): DebertaV2SelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-07, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): DebertaV2Intermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): DebertaV2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-07, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
      (rel_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-07, elementwise_affine=True)
    )
  )
  (pooler): ContextPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): Dropout(p=0, inplace=False)
  )
  (classifier): Linear(in_features=768, out_features=2, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
)
```

[ ]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
deberta.embeddings.word_embeddings.weight requires_grad= True
deberta.embeddings.LayerNorm.weight requires_grad= True
deberta.embeddings.LayerNorm.bias requires_grad= True
deberta.encoder.layer.0.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.0.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.0.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.0.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.0.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.0.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.0.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.0.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.0.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.0.intermediate.dense.bias requires_grad= True
```

deberta.encoder.layer.0.output.dense.weight requires_grad= True
deberta.encoder.layer.0.output.dense.bias requires_grad= True
deberta.encoder.layer.0.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.0.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.1.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.1.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.1.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.1.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.1.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.1.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.1.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.1.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.1.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.1.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.1.output.dense.weight requires_grad= True
deberta.encoder.layer.1.output.dense.bias requires_grad= True
deberta.encoder.layer.1.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.1.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.2.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.2.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.2.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.2.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.2.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.2.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.2.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.2.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.2.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.2.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.2.output.dense.weight requires_grad= True
deberta.encoder.layer.2.output.dense.bias requires_grad= True
deberta.encoder.layer.2.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.2.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.3.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.3.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.3.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.3.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.3.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.3.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.3.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.3.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.3.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.3.intermediate.dense.bias requires_grad= True

deberta.encoder.layer.3.output.dense.weight requires_grad= True
deberta.encoder.layer.3.output.dense.bias requires_grad= True
deberta.encoder.layer.3.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.3.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.4.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.4.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.4.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.4.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.4.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.4.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.4.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.4.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.4.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.4.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.4.output.dense.weight requires_grad= True
deberta.encoder.layer.4.output.dense.bias requires_grad= True
deberta.encoder.layer.4.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.4.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.5.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.5.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.5.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.5.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.5.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.5.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.5.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.5.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.5.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.5.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.5.output.dense.weight requires_grad= True
deberta.encoder.layer.5.output.dense.bias requires_grad= True
deberta.encoder.layer.5.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.5.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.6.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.6.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.6.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.6.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.6.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.6.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.6.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.6.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.6.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.6.intermediate.dense.bias requires_grad= True

deberta.encoder.layer.6.output.dense.weight requires_grad= True
deberta.encoder.layer.6.output.dense.bias requires_grad= True
deberta.encoder.layer.6.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.6.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.7.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.7.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.7.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.7.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.7.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.7.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.7.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.7.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.7.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.7.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.7.output.dense.weight requires_grad= True
deberta.encoder.layer.7.output.dense.bias requires_grad= True
deberta.encoder.layer.7.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.7.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.8.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.8.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.8.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.8.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.8.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.8.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.8.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.8.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.8.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.8.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.8.output.dense.weight requires_grad= True
deberta.encoder.layer.8.output.dense.bias requires_grad= True
deberta.encoder.layer.8.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.8.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.9.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.9.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.9.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.9.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.9.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.9.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.9.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.9.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.9.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.9.intermediate.dense.bias requires_grad= True

```
deberta.encoder.layer.9.output.dense.weight requires_grad= True
deberta.encoder.layer.9.output.dense.bias requires_grad= True
deberta.encoder.layer.9.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.9.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.10.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.10.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.10.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.10.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.10.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.10.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.10.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.10.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.10.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.10.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.10.output.dense.weight requires_grad= True
deberta.encoder.layer.10.output.dense.bias requires_grad= True
deberta.encoder.layer.10.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.10.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.11.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.11.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.11.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.11.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.11.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.11.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.11.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.11.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= True
deberta.encoder.layer.11.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.11.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.11.output.dense.weight requires_grad= True
deberta.encoder.layer.11.output.dense.bias requires_grad= True
deberta.encoder.layer.11.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.11.output.LayerNorm.bias requires_grad= True
deberta.encoder.rel_embeddings.weight requires_grad= True
deberta.encoder.LayerNorm.weight requires_grad= True
deberta.encoder.LayerNorm.bias requires_grad= True
pooler.dense.weight requires_grad= True
pooler.dense.bias requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True
```

[ ]:
```
##########################################
layers_to_unfreeze = [
"deberta.encoder.layer.11.attention.self.query_proj.weight",
```

```
"deberta.encoder.layer.11.attention.self.query_proj.bias",
"deberta.encoder.layer.11.attention.self.key_proj.weight",
"deberta.encoder.layer.11.attention.self.key_proj.bias",
"deberta.encoder.layer.11.attention.self.value_proj.weight",
"deberta.encoder.layer.11.attention.self.value_proj.bias",
"deberta.encoder.layer.11.attention.output.dense.weight",
"deberta.encoder.layer.11.attention.output.dense.bias",
"deberta.encoder.layer.11.attention.output.LayerNorm.weight",
"deberta.encoder.layer.11.attention.output.LayerNorm.bias",
"deberta.encoder.layer.11.intermediate.dense.weight",
"deberta.encoder.layer.11.intermediate.dense.bias",
"deberta.encoder.layer.11.output.dense.weight",
"deberta.encoder.layer.11.output.dense.bias",
"deberta.encoder.layer.11.output.LayerNorm.weight",
"deberta.encoder.layer.11.output.LayerNorm.bias",
"deberta.encoder.rel_embeddings.weight",
"deberta.encoder.LayerNorm.weight",
"deberta.encoder.LayerNorm.bias",
"pooler.dense.weight",
"pooler.dense.bias",
"classifier.weight",
"classifier.bias"
    ]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("=============")
########################################
print("Experiment configuration used with this experiment:")
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
########################################
print("=============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```

```
DebertaV2Config {
  "_attn_implementation_autoset": true,
  "attention_probs_dropout_prob": 0.1,
  "hidden_act": "gelu",
```

```
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-07,
  "legacy": true,
  "max_position_embeddings": 512,
  "max_relative_positions": -1,
  "model_type": "deberta-v2",
  "norm_rel_ebd": "layer_norm",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "pooler_dropout": 0,
  "pooler_hidden_act": "gelu",
  "pooler_hidden_size": 768,
  "pos_att_type": [
    "p2c",
    "c2p"
  ],
  "position_biased_input": false,
  "position_buckets": 256,
  "relative_attention": true,
  "share_att_key": true,
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "type_vocab_size": 0,
  "vocab_size": 128100
}

=============
num_parameters: 184423682
num_trainable_parameters: 8074754
=============
Experiment configuration used with this experiment:
model used: microsoft/deberta-v3-base
learning rate used: 5e-06
number of epochs: 1
maximum sequence length: 128
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
=============
num_trainable_parameters: 8074754
```

```
[ ]: for name, param in model.named_parameters():
         print(name, "requires_grad=", param.requires_grad)
```

```
deberta.embeddings.word_embeddings.weight requires_grad= False
deberta.embeddings.LayerNorm.weight requires_grad= False
deberta.embeddings.LayerNorm.bias requires_grad= False
deberta.encoder.layer.0.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.0.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.0.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.0.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.0.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.0.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.0.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.0.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.0.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.0.attention.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.0.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.0.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.0.output.dense.weight requires_grad= False
deberta.encoder.layer.0.output.dense.bias requires_grad= False
deberta.encoder.layer.0.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.0.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.1.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.1.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.1.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.1.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.1.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.1.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.1.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.1.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.1.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.1.attention.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.1.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.1.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.1.output.dense.weight requires_grad= False
deberta.encoder.layer.1.output.dense.bias requires_grad= False
deberta.encoder.layer.1.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.1.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.2.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.2.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.2.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.2.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.2.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.2.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.2.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.2.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.2.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.2.attention.output.LayerNorm.bias requires_grad= False
```

deberta.encoder.layer.2.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.2.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.2.output.dense.weight requires_grad= False
deberta.encoder.layer.2.output.dense.bias requires_grad= False
deberta.encoder.layer.2.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.2.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.3.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.3.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.3.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.3.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.3.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.3.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.3.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.3.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.3.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.3.attention.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.3.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.3.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.3.output.dense.weight requires_grad= False
deberta.encoder.layer.3.output.dense.bias requires_grad= False
deberta.encoder.layer.3.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.3.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.4.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.4.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.4.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.4.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.4.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.4.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.4.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.4.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.4.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.4.attention.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.4.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.4.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.4.output.dense.weight requires_grad= False
deberta.encoder.layer.4.output.dense.bias requires_grad= False
deberta.encoder.layer.4.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.4.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.5.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.5.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.5.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.5.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.5.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.5.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.5.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.5.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.5.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.5.attention.output.LayerNorm.bias requires_grad= False

deberta.encoder.layer.5.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.5.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.5.output.dense.weight requires_grad= False
deberta.encoder.layer.5.output.dense.bias requires_grad= False
deberta.encoder.layer.5.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.5.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.6.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.6.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.6.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.6.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.6.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.6.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.6.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.6.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.6.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.6.attention.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.6.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.6.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.6.output.dense.weight requires_grad= False
deberta.encoder.layer.6.output.dense.bias requires_grad= False
deberta.encoder.layer.6.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.6.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.7.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.7.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.7.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.7.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.7.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.7.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.7.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.7.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.7.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.7.attention.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.7.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.7.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.7.output.dense.weight requires_grad= False
deberta.encoder.layer.7.output.dense.bias requires_grad= False
deberta.encoder.layer.7.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.7.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.8.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.8.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.8.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.8.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.8.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.8.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.8.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.8.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.8.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.8.attention.output.LayerNorm.bias requires_grad= False

```
deberta.encoder.layer.8.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.8.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.8.output.dense.weight requires_grad= False
deberta.encoder.layer.8.output.dense.bias requires_grad= False
deberta.encoder.layer.8.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.8.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.9.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.9.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.9.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.9.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.9.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.9.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.9.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.9.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.9.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.9.attention.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.9.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.9.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.9.output.dense.weight requires_grad= False
deberta.encoder.layer.9.output.dense.bias requires_grad= False
deberta.encoder.layer.9.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.9.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.10.attention.self.query_proj.weight requires_grad= False
deberta.encoder.layer.10.attention.self.query_proj.bias requires_grad= False
deberta.encoder.layer.10.attention.self.key_proj.weight requires_grad= False
deberta.encoder.layer.10.attention.self.key_proj.bias requires_grad= False
deberta.encoder.layer.10.attention.self.value_proj.weight requires_grad= False
deberta.encoder.layer.10.attention.self.value_proj.bias requires_grad= False
deberta.encoder.layer.10.attention.output.dense.weight requires_grad= False
deberta.encoder.layer.10.attention.output.dense.bias requires_grad= False
deberta.encoder.layer.10.attention.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.10.attention.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.10.intermediate.dense.weight requires_grad= False
deberta.encoder.layer.10.intermediate.dense.bias requires_grad= False
deberta.encoder.layer.10.output.dense.weight requires_grad= False
deberta.encoder.layer.10.output.dense.bias requires_grad= False
deberta.encoder.layer.10.output.LayerNorm.weight requires_grad= False
deberta.encoder.layer.10.output.LayerNorm.bias requires_grad= False
deberta.encoder.layer.11.attention.self.query_proj.weight requires_grad= True
deberta.encoder.layer.11.attention.self.query_proj.bias requires_grad= True
deberta.encoder.layer.11.attention.self.key_proj.weight requires_grad= True
deberta.encoder.layer.11.attention.self.key_proj.bias requires_grad= True
deberta.encoder.layer.11.attention.self.value_proj.weight requires_grad= True
deberta.encoder.layer.11.attention.self.value_proj.bias requires_grad= True
deberta.encoder.layer.11.attention.output.dense.weight requires_grad= True
deberta.encoder.layer.11.attention.output.dense.bias requires_grad= True
deberta.encoder.layer.11.attention.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.11.attention.output.LayerNorm.bias requires_grad= True
```

```
deberta.encoder.layer.11.intermediate.dense.weight requires_grad= True
deberta.encoder.layer.11.intermediate.dense.bias requires_grad= True
deberta.encoder.layer.11.output.dense.weight requires_grad= True
deberta.encoder.layer.11.output.dense.bias requires_grad= True
deberta.encoder.layer.11.output.LayerNorm.weight requires_grad= True
deberta.encoder.layer.11.output.LayerNorm.bias requires_grad= True
deberta.encoder.rel_embeddings.weight requires_grad= True
deberta.encoder.LayerNorm.weight requires_grad= True
deberta.encoder.LayerNorm.bias requires_grad= True
pooler.dense.weight requires_grad= True
pooler.dense.bias requires_grad= True
classifier.weight requires_grad= True
classifier.bias requires_grad= True
```

[ ]:
```
# model.resize_token_embeddings(len(tokenizer))
```

[ ]:
```
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
    val_dataset = val_data_hf,
    output_dir = dir_results,
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-20-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

```
model.safetensors:   0%|          | 0.00/371M [00:00<?, ?B/s]
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 0.6941747069358826, 'eval_accuracy':
0.505938242280285, 'eval_precision': 0.4649122807017544, 'eval_recall':
0.5520833333333334, 'eval_f1': 0.5047619047619047, 'eval_runtime': 1.8294,
```

```
'eval_samples_per_second': 230.135, 'eval_steps_per_second': 2.187, 'epoch':
1.0}
Test metrics: {'eval_loss': 0.691124439239502, 'eval_accuracy':
0.5212649945474372, 'eval_precision': 0.5018867924528302, 'eval_recall':
0.6031746031746031, 'eval_f1': 0.5478887744593203, 'eval_runtime': 2.8591,
'eval_samples_per_second': 320.725, 'eval_steps_per_second': 2.798, 'epoch':
1.0}
```

[ ]:
```
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
  ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to: /content/drive/MyDrive/266-
final/models/single_microsoft/deberta-v3-base_binary_complexity_20250411_121803

<IPython.core.display.HTML object>

EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```

0.2.10 snc xlnet/xlnet-base-cased regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[93]:
```
# Define Experiment Parameters
# named_model = "bert-base-cased"
# named_model = "roberta-base"
# named_model = "bert-large-cased"
# named_model = "roberta-large"
# named_model = "answerdotai/ModernBERT-base" # modern bert
# named_model = "answerdotai/ModernBERT-large" # modern bert
# named_model = "microsoft/deberta-v3-base" # deberta
named_model = "xlnet/xlnet-base-cased" #
# named_model = "xlnet/xlnet-large-cased" #
#############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
num_epochs = 1
###########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
#############
y_col = "binary_complexity"
# y_col = "complexity"
#############
x_task = "single"
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
###########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
```

```
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
##########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="xlnet/xlnet-base-cased",
    local_model_path=None,
    config=None)
############
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
#     local_model_path="...CONFIGURE_PATH...",
#     config=custom_config)
##########################################
print("=============")
print(named_model, ":")
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters at load:", model.
 ↪num_parameters(only_trainable=True))
print("=============")
print("model lineage:", MODEL_LINEAGE)
print("=============")
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]

Map:   0%|          | 0/421 [00:00<?, ? examples/s]

Map:   0%|          | 0/917 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([  101,  1252,  1106,  1103,  3824,
1104, 19892, 11220,  1324,  1119,
         1522,  3839,   117,  1272,  1103,  1555,  1104,  1103, 11563,  5609,
         1106,  1172,   132,  1152,  2446,  1122,  1113,  1147,  3221,   119,
          102,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
```

```
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0]),
'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0])}
Loading from Hugging Face model: xlnet/xlnet-base-cased

config.json:   0%|          | 0.00/760 [00:00<?, ?B/s]

spiece.model:   0%|          | 0.00/798k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.38M [00:00<?, ?B/s]

pytorch_model.bin:   0%|          | 0.00/467M [00:00<?, ?B/s]

Some weights of XLNetForSequenceClassification were not initialized from the
model checkpoint at xlnet/xlnet-base-cased and are newly initialized:
['logits_proj.bias', 'logits_proj.weight', 'sequence_summary.summary.bias',
'sequence_summary.summary.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

=============
xlnet/xlnet-base-cased :
=============
num_parameters: 117310466
num_trainable_parameters at load: 117310466
=============
model lineage: {'type': 'huggingface_hub', 'path': 'xlnet/xlnet-base-cased',
'timestamp': '2025-04-11 12:22:26'}
=============
```

[94]:
```
print(model)
```

```
XLNetForSequenceClassification(
  (transformer): XLNetModel(
    (word_embedding): Embedding(32000, 768)
    (layer): ModuleList(
      (0-11): 12 x XLNetLayer(
        (rel_attn): XLNetRelativeAttention(
          (layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
```

```
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (ff): XLNetFeedForward(
          (layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (layer_1): Linear(in_features=768, out_features=3072, bias=True)
          (layer_2): Linear(in_features=3072, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
          (activation_function): GELUActivation()
        )
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (sequence_summary): SequenceSummary(
    (summary): Linear(in_features=768, out_features=768, bias=True)
    (activation): Tanh()
    (first_dropout): Identity()
    (last_dropout): Dropout(p=0.1, inplace=False)
  )
  (logits_proj): Linear(in_features=768, out_features=2, bias=True)
)
```

[95]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
transformer.mask_emb requires_grad= True
transformer.word_embedding.weight requires_grad= True
transformer.layer.0.rel_attn.q requires_grad= True
transformer.layer.0.rel_attn.k requires_grad= True
transformer.layer.0.rel_attn.v requires_grad= True
transformer.layer.0.rel_attn.o requires_grad= True
transformer.layer.0.rel_attn.r requires_grad= True
transformer.layer.0.rel_attn.r_r_bias requires_grad= True
transformer.layer.0.rel_attn.r_s_bias requires_grad= True
transformer.layer.0.rel_attn.r_w_bias requires_grad= True
transformer.layer.0.rel_attn.seg_embed requires_grad= True
transformer.layer.0.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.0.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.0.ff.layer_norm.weight requires_grad= True
transformer.layer.0.ff.layer_norm.bias requires_grad= True
transformer.layer.0.ff.layer_1.weight requires_grad= True
transformer.layer.0.ff.layer_1.bias requires_grad= True
transformer.layer.0.ff.layer_2.weight requires_grad= True
transformer.layer.0.ff.layer_2.bias requires_grad= True
transformer.layer.1.rel_attn.q requires_grad= True
transformer.layer.1.rel_attn.k requires_grad= True
transformer.layer.1.rel_attn.v requires_grad= True
```

```
transformer.layer.1.rel_attn.o requires_grad= True
transformer.layer.1.rel_attn.r requires_grad= True
transformer.layer.1.rel_attn.r_r_bias requires_grad= True
transformer.layer.1.rel_attn.r_s_bias requires_grad= True
transformer.layer.1.rel_attn.r_w_bias requires_grad= True
transformer.layer.1.rel_attn.seg_embed requires_grad= True
transformer.layer.1.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.1.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.1.ff.layer_norm.weight requires_grad= True
transformer.layer.1.ff.layer_norm.bias requires_grad= True
transformer.layer.1.ff.layer_1.weight requires_grad= True
transformer.layer.1.ff.layer_1.bias requires_grad= True
transformer.layer.1.ff.layer_2.weight requires_grad= True
transformer.layer.1.ff.layer_2.bias requires_grad= True
transformer.layer.2.rel_attn.q requires_grad= True
transformer.layer.2.rel_attn.k requires_grad= True
transformer.layer.2.rel_attn.v requires_grad= True
transformer.layer.2.rel_attn.o requires_grad= True
transformer.layer.2.rel_attn.r requires_grad= True
transformer.layer.2.rel_attn.r_r_bias requires_grad= True
transformer.layer.2.rel_attn.r_s_bias requires_grad= True
transformer.layer.2.rel_attn.r_w_bias requires_grad= True
transformer.layer.2.rel_attn.seg_embed requires_grad= True
transformer.layer.2.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.2.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.2.ff.layer_norm.weight requires_grad= True
transformer.layer.2.ff.layer_norm.bias requires_grad= True
transformer.layer.2.ff.layer_1.weight requires_grad= True
transformer.layer.2.ff.layer_1.bias requires_grad= True
transformer.layer.2.ff.layer_2.weight requires_grad= True
transformer.layer.2.ff.layer_2.bias requires_grad= True
transformer.layer.3.rel_attn.q requires_grad= True
transformer.layer.3.rel_attn.k requires_grad= True
transformer.layer.3.rel_attn.v requires_grad= True
transformer.layer.3.rel_attn.o requires_grad= True
transformer.layer.3.rel_attn.r requires_grad= True
transformer.layer.3.rel_attn.r_r_bias requires_grad= True
transformer.layer.3.rel_attn.r_s_bias requires_grad= True
transformer.layer.3.rel_attn.r_w_bias requires_grad= True
transformer.layer.3.rel_attn.seg_embed requires_grad= True
transformer.layer.3.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.3.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.3.ff.layer_norm.weight requires_grad= True
transformer.layer.3.ff.layer_norm.bias requires_grad= True
transformer.layer.3.ff.layer_1.weight requires_grad= True
transformer.layer.3.ff.layer_1.bias requires_grad= True
transformer.layer.3.ff.layer_2.weight requires_grad= True
transformer.layer.3.ff.layer_2.bias requires_grad= True
```

```
transformer.layer.4.rel_attn.q requires_grad= True
transformer.layer.4.rel_attn.k requires_grad= True
transformer.layer.4.rel_attn.v requires_grad= True
transformer.layer.4.rel_attn.o requires_grad= True
transformer.layer.4.rel_attn.r requires_grad= True
transformer.layer.4.rel_attn.r_r_bias requires_grad= True
transformer.layer.4.rel_attn.r_s_bias requires_grad= True
transformer.layer.4.rel_attn.r_w_bias requires_grad= True
transformer.layer.4.rel_attn.seg_embed requires_grad= True
transformer.layer.4.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.4.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.4.ff.layer_norm.weight requires_grad= True
transformer.layer.4.ff.layer_norm.bias requires_grad= True
transformer.layer.4.ff.layer_1.weight requires_grad= True
transformer.layer.4.ff.layer_1.bias requires_grad= True
transformer.layer.4.ff.layer_2.weight requires_grad= True
transformer.layer.4.ff.layer_2.bias requires_grad= True
transformer.layer.5.rel_attn.q requires_grad= True
transformer.layer.5.rel_attn.k requires_grad= True
transformer.layer.5.rel_attn.v requires_grad= True
transformer.layer.5.rel_attn.o requires_grad= True
transformer.layer.5.rel_attn.r requires_grad= True
transformer.layer.5.rel_attn.r_r_bias requires_grad= True
transformer.layer.5.rel_attn.r_s_bias requires_grad= True
transformer.layer.5.rel_attn.r_w_bias requires_grad= True
transformer.layer.5.rel_attn.seg_embed requires_grad= True
transformer.layer.5.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.5.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.5.ff.layer_norm.weight requires_grad= True
transformer.layer.5.ff.layer_norm.bias requires_grad= True
transformer.layer.5.ff.layer_1.weight requires_grad= True
transformer.layer.5.ff.layer_1.bias requires_grad= True
transformer.layer.5.ff.layer_2.weight requires_grad= True
transformer.layer.5.ff.layer_2.bias requires_grad= True
transformer.layer.6.rel_attn.q requires_grad= True
transformer.layer.6.rel_attn.k requires_grad= True
transformer.layer.6.rel_attn.v requires_grad= True
transformer.layer.6.rel_attn.o requires_grad= True
transformer.layer.6.rel_attn.r requires_grad= True
transformer.layer.6.rel_attn.r_r_bias requires_grad= True
transformer.layer.6.rel_attn.r_s_bias requires_grad= True
transformer.layer.6.rel_attn.r_w_bias requires_grad= True
transformer.layer.6.rel_attn.seg_embed requires_grad= True
transformer.layer.6.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.6.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.6.ff.layer_norm.weight requires_grad= True
transformer.layer.6.ff.layer_norm.bias requires_grad= True
transformer.layer.6.ff.layer_1.weight requires_grad= True
```

```
transformer.layer.6.ff.layer_1.bias requires_grad= True
transformer.layer.6.ff.layer_2.weight requires_grad= True
transformer.layer.6.ff.layer_2.bias requires_grad= True
transformer.layer.7.rel_attn.q requires_grad= True
transformer.layer.7.rel_attn.k requires_grad= True
transformer.layer.7.rel_attn.v requires_grad= True
transformer.layer.7.rel_attn.o requires_grad= True
transformer.layer.7.rel_attn.r requires_grad= True
transformer.layer.7.rel_attn.r_r_bias requires_grad= True
transformer.layer.7.rel_attn.r_s_bias requires_grad= True
transformer.layer.7.rel_attn.r_w_bias requires_grad= True
transformer.layer.7.rel_attn.seg_embed requires_grad= True
transformer.layer.7.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.7.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.7.ff.layer_norm.weight requires_grad= True
transformer.layer.7.ff.layer_norm.bias requires_grad= True
transformer.layer.7.ff.layer_1.weight requires_grad= True
transformer.layer.7.ff.layer_1.bias requires_grad= True
transformer.layer.7.ff.layer_2.weight requires_grad= True
transformer.layer.7.ff.layer_2.bias requires_grad= True
transformer.layer.8.rel_attn.q requires_grad= True
transformer.layer.8.rel_attn.k requires_grad= True
transformer.layer.8.rel_attn.v requires_grad= True
transformer.layer.8.rel_attn.o requires_grad= True
transformer.layer.8.rel_attn.r requires_grad= True
transformer.layer.8.rel_attn.r_r_bias requires_grad= True
transformer.layer.8.rel_attn.r_s_bias requires_grad= True
transformer.layer.8.rel_attn.r_w_bias requires_grad= True
transformer.layer.8.rel_attn.seg_embed requires_grad= True
transformer.layer.8.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.8.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.8.ff.layer_norm.weight requires_grad= True
transformer.layer.8.ff.layer_norm.bias requires_grad= True
transformer.layer.8.ff.layer_1.weight requires_grad= True
transformer.layer.8.ff.layer_1.bias requires_grad= True
transformer.layer.8.ff.layer_2.weight requires_grad= True
transformer.layer.8.ff.layer_2.bias requires_grad= True
transformer.layer.9.rel_attn.q requires_grad= True
transformer.layer.9.rel_attn.k requires_grad= True
transformer.layer.9.rel_attn.v requires_grad= True
transformer.layer.9.rel_attn.o requires_grad= True
transformer.layer.9.rel_attn.r requires_grad= True
transformer.layer.9.rel_attn.r_r_bias requires_grad= True
transformer.layer.9.rel_attn.r_s_bias requires_grad= True
transformer.layer.9.rel_attn.r_w_bias requires_grad= True
transformer.layer.9.rel_attn.seg_embed requires_grad= True
transformer.layer.9.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.9.rel_attn.layer_norm.bias requires_grad= True
```

```
transformer.layer.9.ff.layer_norm.weight requires_grad= True
transformer.layer.9.ff.layer_norm.bias requires_grad= True
transformer.layer.9.ff.layer_1.weight requires_grad= True
transformer.layer.9.ff.layer_1.bias requires_grad= True
transformer.layer.9.ff.layer_2.weight requires_grad= True
transformer.layer.9.ff.layer_2.bias requires_grad= True
transformer.layer.10.rel_attn.q requires_grad= True
transformer.layer.10.rel_attn.k requires_grad= True
transformer.layer.10.rel_attn.v requires_grad= True
transformer.layer.10.rel_attn.o requires_grad= True
transformer.layer.10.rel_attn.r requires_grad= True
transformer.layer.10.rel_attn.r_r_bias requires_grad= True
transformer.layer.10.rel_attn.r_s_bias requires_grad= True
transformer.layer.10.rel_attn.r_w_bias requires_grad= True
transformer.layer.10.rel_attn.seg_embed requires_grad= True
transformer.layer.10.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.10.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.10.ff.layer_norm.weight requires_grad= True
transformer.layer.10.ff.layer_norm.bias requires_grad= True
transformer.layer.10.ff.layer_1.weight requires_grad= True
transformer.layer.10.ff.layer_1.bias requires_grad= True
transformer.layer.10.ff.layer_2.weight requires_grad= True
transformer.layer.10.ff.layer_2.bias requires_grad= True
transformer.layer.11.rel_attn.q requires_grad= True
transformer.layer.11.rel_attn.k requires_grad= True
transformer.layer.11.rel_attn.v requires_grad= True
transformer.layer.11.rel_attn.o requires_grad= True
transformer.layer.11.rel_attn.r requires_grad= True
transformer.layer.11.rel_attn.r_r_bias requires_grad= True
transformer.layer.11.rel_attn.r_s_bias requires_grad= True
transformer.layer.11.rel_attn.r_w_bias requires_grad= True
transformer.layer.11.rel_attn.seg_embed requires_grad= True
transformer.layer.11.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.11.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.11.ff.layer_norm.weight requires_grad= True
transformer.layer.11.ff.layer_norm.bias requires_grad= True
transformer.layer.11.ff.layer_1.weight requires_grad= True
transformer.layer.11.ff.layer_1.bias requires_grad= True
transformer.layer.11.ff.layer_2.weight requires_grad= True
transformer.layer.11.ff.layer_2.bias requires_grad= True
sequence_summary.summary.weight requires_grad= True
sequence_summary.summary.bias requires_grad= True
logits_proj.weight requires_grad= True
logits_proj.bias requires_grad= True
```

[96]:
```
##########################################
layers_to_unfreeze = [
```

```
"transformer.layer.11.rel_attn.q",
"transformer.layer.11.rel_attn.k",
"transformer.layer.11.rel_attn.v",
"transformer.layer.11.rel_attn.o",
"transformer.layer.11.rel_attn.r",
"transformer.layer.11.rel_attn.r_r_bias",
"transformer.layer.11.rel_attn.r_s_bias",
"transformer.layer.11.rel_attn.r_w_bias",
"transformer.layer.11.rel_attn.seg_embed",
"transformer.layer.11.rel_attn.layer_norm.weight",
"transformer.layer.11.rel_attn.layer_norm.bias",
"transformer.layer.11.ff.layer_norm.weight",
"transformer.layer.11.ff.layer_norm.bias",
"transformer.layer.11.ff.layer_1.weight",
"transformer.layer.11.ff.layer_1.bias",
"transformer.layer.11.ff.layer_2.weight",
"transformer.layer.11.ff.layer_2.bias",
"sequence_summary.summary.weight",
"sequence_summary.summary.bias",
"logits_proj.weight",
"logits_proj.bias"
    ]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("=============")
##########################################
print("Experiment configuration used with this experiment:")
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
##########################################
print("=============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```

```
XLNetConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "XLNetLMHeadModel"
  ],
```

```
  "attn_type": "bi",
  "bi_data": false,
  "bos_token_id": 1,
  "clamp_len": -1,
  "d_head": 64,
  "d_inner": 3072,
  "d_model": 768,
  "dropout": 0.1,
  "end_n_top": 5,
  "eos_token_id": 2,
  "ff_activation": "gelu",
  "initializer_range": 0.02,
  "layer_norm_eps": 1e-12,
  "mem_len": null,
  "model_type": "xlnet",
  "n_head": 12,
  "n_layer": 12,
  "pad_token_id": 5,
  "reuse_len": null,
  "same_length": false,
  "start_n_top": 5,
  "summary_activation": "tanh",
  "summary_last_dropout": 0.1,
  "summary_type": "last",
  "summary_use_proj": true,
  "task_specific_params": {
    "text-generation": {
      "do_sample": true,
      "max_length": 250
    }
  },
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "untie_r": true,
  "use_mems_eval": true,
  "use_mems_train": false,
  "vocab_size": 32000
}

=============
num_parameters: 117310466
num_trainable_parameters: 8270594
=============
Experiment configuration used with this experiment:
model used: xlnet/xlnet-base-cased
learning rate used: 5e-06
number of epochs: 1
maximum sequence length: 128
```

```
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
=============
num_trainable_parameters: 8270594
```

[97]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
transformer.mask_emb requires_grad= False
transformer.word_embedding.weight requires_grad= False
transformer.layer.0.rel_attn.q requires_grad= False
transformer.layer.0.rel_attn.k requires_grad= False
transformer.layer.0.rel_attn.v requires_grad= False
transformer.layer.0.rel_attn.o requires_grad= False
transformer.layer.0.rel_attn.r requires_grad= False
transformer.layer.0.rel_attn.r_r_bias requires_grad= False
transformer.layer.0.rel_attn.r_s_bias requires_grad= False
transformer.layer.0.rel_attn.r_w_bias requires_grad= False
transformer.layer.0.rel_attn.seg_embed requires_grad= False
transformer.layer.0.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.0.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.0.ff.layer_norm.weight requires_grad= False
transformer.layer.0.ff.layer_norm.bias requires_grad= False
transformer.layer.0.ff.layer_1.weight requires_grad= False
transformer.layer.0.ff.layer_1.bias requires_grad= False
transformer.layer.0.ff.layer_2.weight requires_grad= False
transformer.layer.0.ff.layer_2.bias requires_grad= False
transformer.layer.1.rel_attn.q requires_grad= False
transformer.layer.1.rel_attn.k requires_grad= False
transformer.layer.1.rel_attn.v requires_grad= False
transformer.layer.1.rel_attn.o requires_grad= False
transformer.layer.1.rel_attn.r requires_grad= False
transformer.layer.1.rel_attn.r_r_bias requires_grad= False
transformer.layer.1.rel_attn.r_s_bias requires_grad= False
transformer.layer.1.rel_attn.r_w_bias requires_grad= False
transformer.layer.1.rel_attn.seg_embed requires_grad= False
transformer.layer.1.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.1.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.1.ff.layer_norm.weight requires_grad= False
transformer.layer.1.ff.layer_norm.bias requires_grad= False
transformer.layer.1.ff.layer_1.weight requires_grad= False
transformer.layer.1.ff.layer_1.bias requires_grad= False
transformer.layer.1.ff.layer_2.weight requires_grad= False
transformer.layer.1.ff.layer_2.bias requires_grad= False
transformer.layer.2.rel_attn.q requires_grad= False
```

```
transformer.layer.2.rel_attn.k requires_grad= False
transformer.layer.2.rel_attn.v requires_grad= False
transformer.layer.2.rel_attn.o requires_grad= False
transformer.layer.2.rel_attn.r requires_grad= False
transformer.layer.2.rel_attn.r_r_bias requires_grad= False
transformer.layer.2.rel_attn.r_s_bias requires_grad= False
transformer.layer.2.rel_attn.r_w_bias requires_grad= False
transformer.layer.2.rel_attn.seg_embed requires_grad= False
transformer.layer.2.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.2.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.2.ff.layer_norm.weight requires_grad= False
transformer.layer.2.ff.layer_norm.bias requires_grad= False
transformer.layer.2.ff.layer_1.weight requires_grad= False
transformer.layer.2.ff.layer_1.bias requires_grad= False
transformer.layer.2.ff.layer_2.weight requires_grad= False
transformer.layer.2.ff.layer_2.bias requires_grad= False
transformer.layer.3.rel_attn.q requires_grad= False
transformer.layer.3.rel_attn.k requires_grad= False
transformer.layer.3.rel_attn.v requires_grad= False
transformer.layer.3.rel_attn.o requires_grad= False
transformer.layer.3.rel_attn.r requires_grad= False
transformer.layer.3.rel_attn.r_r_bias requires_grad= False
transformer.layer.3.rel_attn.r_s_bias requires_grad= False
transformer.layer.3.rel_attn.r_w_bias requires_grad= False
transformer.layer.3.rel_attn.seg_embed requires_grad= False
transformer.layer.3.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.3.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.3.ff.layer_norm.weight requires_grad= False
transformer.layer.3.ff.layer_norm.bias requires_grad= False
transformer.layer.3.ff.layer_1.weight requires_grad= False
transformer.layer.3.ff.layer_1.bias requires_grad= False
transformer.layer.3.ff.layer_2.weight requires_grad= False
transformer.layer.3.ff.layer_2.bias requires_grad= False
transformer.layer.4.rel_attn.q requires_grad= False
transformer.layer.4.rel_attn.k requires_grad= False
transformer.layer.4.rel_attn.v requires_grad= False
transformer.layer.4.rel_attn.o requires_grad= False
transformer.layer.4.rel_attn.r requires_grad= False
transformer.layer.4.rel_attn.r_r_bias requires_grad= False
transformer.layer.4.rel_attn.r_s_bias requires_grad= False
transformer.layer.4.rel_attn.r_w_bias requires_grad= False
transformer.layer.4.rel_attn.seg_embed requires_grad= False
transformer.layer.4.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.4.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.4.ff.layer_norm.weight requires_grad= False
transformer.layer.4.ff.layer_norm.bias requires_grad= False
transformer.layer.4.ff.layer_1.weight requires_grad= False
transformer.layer.4.ff.layer_1.bias requires_grad= False
```

```
transformer.layer.4.ff.layer_2.weight requires_grad= False
transformer.layer.4.ff.layer_2.bias requires_grad= False
transformer.layer.5.rel_attn.q requires_grad= False
transformer.layer.5.rel_attn.k requires_grad= False
transformer.layer.5.rel_attn.v requires_grad= False
transformer.layer.5.rel_attn.o requires_grad= False
transformer.layer.5.rel_attn.r requires_grad= False
transformer.layer.5.rel_attn.r_r_bias requires_grad= False
transformer.layer.5.rel_attn.r_s_bias requires_grad= False
transformer.layer.5.rel_attn.r_w_bias requires_grad= False
transformer.layer.5.rel_attn.seg_embed requires_grad= False
transformer.layer.5.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.5.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.5.ff.layer_norm.weight requires_grad= False
transformer.layer.5.ff.layer_norm.bias requires_grad= False
transformer.layer.5.ff.layer_1.weight requires_grad= False
transformer.layer.5.ff.layer_1.bias requires_grad= False
transformer.layer.5.ff.layer_2.weight requires_grad= False
transformer.layer.5.ff.layer_2.bias requires_grad= False
transformer.layer.6.rel_attn.q requires_grad= False
transformer.layer.6.rel_attn.k requires_grad= False
transformer.layer.6.rel_attn.v requires_grad= False
transformer.layer.6.rel_attn.o requires_grad= False
transformer.layer.6.rel_attn.r requires_grad= False
transformer.layer.6.rel_attn.r_r_bias requires_grad= False
transformer.layer.6.rel_attn.r_s_bias requires_grad= False
transformer.layer.6.rel_attn.r_w_bias requires_grad= False
transformer.layer.6.rel_attn.seg_embed requires_grad= False
transformer.layer.6.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.6.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.6.ff.layer_norm.weight requires_grad= False
transformer.layer.6.ff.layer_norm.bias requires_grad= False
transformer.layer.6.ff.layer_1.weight requires_grad= False
transformer.layer.6.ff.layer_1.bias requires_grad= False
transformer.layer.6.ff.layer_2.weight requires_grad= False
transformer.layer.6.ff.layer_2.bias requires_grad= False
transformer.layer.7.rel_attn.q requires_grad= False
transformer.layer.7.rel_attn.k requires_grad= False
transformer.layer.7.rel_attn.v requires_grad= False
transformer.layer.7.rel_attn.o requires_grad= False
transformer.layer.7.rel_attn.r requires_grad= False
transformer.layer.7.rel_attn.r_r_bias requires_grad= False
transformer.layer.7.rel_attn.r_s_bias requires_grad= False
transformer.layer.7.rel_attn.r_w_bias requires_grad= False
transformer.layer.7.rel_attn.seg_embed requires_grad= False
transformer.layer.7.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.7.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.7.ff.layer_norm.weight requires_grad= False
```

```
transformer.layer.7.ff.layer_norm.bias requires_grad= False
transformer.layer.7.ff.layer_1.weight requires_grad= False
transformer.layer.7.ff.layer_1.bias requires_grad= False
transformer.layer.7.ff.layer_2.weight requires_grad= False
transformer.layer.7.ff.layer_2.bias requires_grad= False
transformer.layer.8.rel_attn.q requires_grad= False
transformer.layer.8.rel_attn.k requires_grad= False
transformer.layer.8.rel_attn.v requires_grad= False
transformer.layer.8.rel_attn.o requires_grad= False
transformer.layer.8.rel_attn.r requires_grad= False
transformer.layer.8.rel_attn.r_r_bias requires_grad= False
transformer.layer.8.rel_attn.r_s_bias requires_grad= False
transformer.layer.8.rel_attn.r_w_bias requires_grad= False
transformer.layer.8.rel_attn.seg_embed requires_grad= False
transformer.layer.8.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.8.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.8.ff.layer_norm.weight requires_grad= False
transformer.layer.8.ff.layer_norm.bias requires_grad= False
transformer.layer.8.ff.layer_1.weight requires_grad= False
transformer.layer.8.ff.layer_1.bias requires_grad= False
transformer.layer.8.ff.layer_2.weight requires_grad= False
transformer.layer.8.ff.layer_2.bias requires_grad= False
transformer.layer.9.rel_attn.q requires_grad= False
transformer.layer.9.rel_attn.k requires_grad= False
transformer.layer.9.rel_attn.v requires_grad= False
transformer.layer.9.rel_attn.o requires_grad= False
transformer.layer.9.rel_attn.r requires_grad= False
transformer.layer.9.rel_attn.r_r_bias requires_grad= False
transformer.layer.9.rel_attn.r_s_bias requires_grad= False
transformer.layer.9.rel_attn.r_w_bias requires_grad= False
transformer.layer.9.rel_attn.seg_embed requires_grad= False
transformer.layer.9.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.9.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.9.ff.layer_norm.weight requires_grad= False
transformer.layer.9.ff.layer_norm.bias requires_grad= False
transformer.layer.9.ff.layer_1.weight requires_grad= False
transformer.layer.9.ff.layer_1.bias requires_grad= False
transformer.layer.9.ff.layer_2.weight requires_grad= False
transformer.layer.9.ff.layer_2.bias requires_grad= False
transformer.layer.10.rel_attn.q requires_grad= False
transformer.layer.10.rel_attn.k requires_grad= False
transformer.layer.10.rel_attn.v requires_grad= False
transformer.layer.10.rel_attn.o requires_grad= False
transformer.layer.10.rel_attn.r requires_grad= False
transformer.layer.10.rel_attn.r_r_bias requires_grad= False
transformer.layer.10.rel_attn.r_s_bias requires_grad= False
transformer.layer.10.rel_attn.r_w_bias requires_grad= False
transformer.layer.10.rel_attn.seg_embed requires_grad= False
```

```
transformer.layer.10.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.10.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.10.ff.layer_norm.weight requires_grad= False
transformer.layer.10.ff.layer_norm.bias requires_grad= False
transformer.layer.10.ff.layer_1.weight requires_grad= False
transformer.layer.10.ff.layer_1.bias requires_grad= False
transformer.layer.10.ff.layer_2.weight requires_grad= False
transformer.layer.10.ff.layer_2.bias requires_grad= False
transformer.layer.11.rel_attn.q requires_grad= True
transformer.layer.11.rel_attn.k requires_grad= True
transformer.layer.11.rel_attn.v requires_grad= True
transformer.layer.11.rel_attn.o requires_grad= True
transformer.layer.11.rel_attn.r requires_grad= True
transformer.layer.11.rel_attn.r_r_bias requires_grad= True
transformer.layer.11.rel_attn.r_s_bias requires_grad= True
transformer.layer.11.rel_attn.r_w_bias requires_grad= True
transformer.layer.11.rel_attn.seg_embed requires_grad= True
transformer.layer.11.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.11.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.11.ff.layer_norm.weight requires_grad= True
transformer.layer.11.ff.layer_norm.bias requires_grad= True
transformer.layer.11.ff.layer_1.weight requires_grad= True
transformer.layer.11.ff.layer_1.bias requires_grad= True
transformer.layer.11.ff.layer_2.weight requires_grad= True
transformer.layer.11.ff.layer_2.bias requires_grad= True
sequence_summary.summary.weight requires_grad= True
sequence_summary.summary.bias requires_grad= True
logits_proj.weight requires_grad= True
logits_proj.bias requires_grad= True
```

[98]:
```
# model.resize_token_embeddings(len(tokenizer))
```

[99]:
```
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
    val_dataset = val_data_hf,
    output_dir = dir_results,
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-88-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

```
model.safetensors:   0%|          | 0.00/467M [00:00<?, ?B/s]
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 0.6938373446464539, 'eval_accuracy':
0.49643705463182897, 'eval_precision': 0.4494949494949495, 'eval_recall':
0.4635416666666667, 'eval_f1': 0.4564102564102564, 'eval_runtime': 2.8709,
'eval_samples_per_second': 146.645, 'eval_steps_per_second': 1.393, 'epoch':
1.0}
Test metrics: {'eval_loss': 0.6911154389381409, 'eval_accuracy':
0.5376226826608506, 'eval_precision': 0.5192743764172335, 'eval_recall':
0.5192743764172335, 'eval_f1': 0.5192743764172335, 'eval_runtime': 3.3129,
'eval_samples_per_second': 276.795, 'eval_steps_per_second': 2.415, 'epoch':
1.0}
```

[100]:
```
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
  ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
```

```
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to:
/content/drive/MyDrive/266-final/models/single_xlnet/xlnet-base-
cased_binary_complexity_20250411_122256

<IPython.core.display.HTML object>

EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```

[ ]:

0.2.11 snc xlnet/xlnet-large-cased regularization_weight_decay = 0.5 learning_rate = 5e-6 size_batch = 128 length_max = 128 num_epochs = 1

[101]:
```
# Define Experiment Parameters
# named_model = "bert-base-cased"
# named_model = "roberta-base"
# named_model = "bert-large-cased"
# named_model = "roberta-large"
# named_model = "answerdotai/ModernBERT-base" # modern bert
# named_model = "answerdotai/ModernBERT-large" # modern bert
# named_model = "microsoft/deberta-v3-base" # deberta
# named_model = "xlnet/xlnet-base-cased" #
named_model = "xlnet/xlnet-large-cased" #
#############
regularization_weight_decay = 0.5
learning_rate = 5e-6
size_batch = 128
length_max = 128
num_epochs = 1
##########################################
# x_col = "sentence"
x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
#############
y_col = "binary_complexity"
# y_col = "complexity"
#############
x_task = "single"
```

```
# x_task = "multi"
if x_task == "single":
    df_train = train_single_df
    df_val   = trial_val_single_df
    df_test  = test_single_df
else:
    df_train = train_multi_df
    df_val   = trial_val_multi_df
    df_test  = test_multi_df
########################################
# Tokenize & Prepare Datasets
train_data_hf = prepare_dataset(
    df_train,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
val_data_hf = prepare_dataset(
    df_val,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
test_data_hf = prepare_dataset(
    df_test,
    tokenizer,
    text_col=x_col,
    label_col=y_col,
    max_length=length_max)
print("Datasets prepared. Sample from train_data_hf:\n", train_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", val_data_hf[10])
# print("Datasets prepared. Sample from train_data_hf:\n", test_data_hf[10])
########################################
model, tokenizer = get_model_and_tokenizer(
    remote_model_name="xlnet/xlnet-large-cased",
    local_model_path=None,
    config=None)
############
# model, tokenizer = get_model_and_tokenizer(
#     remote_model_name=None
#     local_model_path="...CONFIGURE_PATH...",
#     config=custom_config)
########################################
print("==============")
print(named_model, ":")
print("==============")
print("num_parameters:", model.num_parameters())
```

```
print("num_trainable_parameters at load:", model.
  ↪num_parameters(only_trainable=True))
print("==============")
print("model lineage:", MODEL_LINEAGE)
print("==============")
```

```
Map:   0%|          | 0/7662 [00:00<?, ? examples/s]

Map:   0%|          | 0/421 [00:00<?, ? examples/s]

Map:   0%|          | 0/917 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([    5,     5,     5,     5,     5,
5,     5,     5,     5,     5,
            5,     5,     5,     5,     5,     5,     5,     5,     5,     5,
            5,     5,     5,     5,     5,     5,     5,     5,     5,     5,
            5,     5,     5,     5,     5,     5,     5,     5,     5,     5,
            5,     5,     5,     5,     5,     5,     5,     5,     5,     5,
            5,     5,     5,     5,     5,     5,     5,     5,     5,     5,
            5,     5,     5,     5,     5,     5,     5,     5,     5,     5,
            5,     5,     5,     5,     5,     5,     5,     5,   130,    22,
           18,  5297,    20, 20289,  5635,    43,   675,  3183,    19,   149,
           18,   348,    20,    18, 14712,  8762,    22,   107,    97,    63,
         1708,    36,    31,    58,  5466,     9,     4,     3]),
'attention_mask': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1])}
Loading from Hugging Face model: xlnet/xlnet-large-cased

config.json:   0%|          | 0.00/761 [00:00<?, ?B/s]

spiece.model:   0%|          | 0.00/798k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.38M [00:00<?, ?B/s]

pytorch_model.bin:   0%|          | 0.00/1.44G [00:00<?, ?B/s]

Some weights of XLNetForSequenceClassification were not initialized from the
model checkpoint at xlnet/xlnet-large-cased and are newly initialized:
['logits_proj.bias', 'logits_proj.weight', 'sequence_summary.summary.bias',
'sequence_summary.summary.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

==============
```

```
xlnet/xlnet-large-cased :
=============
num_parameters: 361320450
num_trainable_parameters at load: 361320450
=============
model lineage: {'type': 'huggingface_hub', 'path': 'xlnet/xlnet-large-cased',
'timestamp': '2025-04-11 12:23:34'}
=============
```

[102]:
```
print(model)
```

```
XLNetForSequenceClassification(
  (transformer): XLNetModel(
    (word_embedding): Embedding(32000, 1024)
    (layer): ModuleList(
      (0-23): 24 x XLNetLayer(
        (rel_attn): XLNetRelativeAttention(
          (layer_norm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (ff): XLNetFeedForward(
          (layer_norm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
          (layer_1): Linear(in_features=1024, out_features=4096, bias=True)
          (layer_2): Linear(in_features=4096, out_features=1024, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
          (activation_function): GELUActivation()
        )
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (sequence_summary): SequenceSummary(
    (summary): Linear(in_features=1024, out_features=1024, bias=True)
    (activation): Tanh()
    (first_dropout): Identity()
    (last_dropout): Dropout(p=0.1, inplace=False)
  )
  (logits_proj): Linear(in_features=1024, out_features=2, bias=True)
)
```

[103]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
transformer.mask_emb requires_grad= True
transformer.word_embedding.weight requires_grad= True
transformer.layer.0.rel_attn.q requires_grad= True
transformer.layer.0.rel_attn.k requires_grad= True
transformer.layer.0.rel_attn.v requires_grad= True
```

```
transformer.layer.0.rel_attn.o requires_grad= True
transformer.layer.0.rel_attn.r requires_grad= True
transformer.layer.0.rel_attn.r_r_bias requires_grad= True
transformer.layer.0.rel_attn.r_s_bias requires_grad= True
transformer.layer.0.rel_attn.r_w_bias requires_grad= True
transformer.layer.0.rel_attn.seg_embed requires_grad= True
transformer.layer.0.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.0.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.0.ff.layer_norm.weight requires_grad= True
transformer.layer.0.ff.layer_norm.bias requires_grad= True
transformer.layer.0.ff.layer_1.weight requires_grad= True
transformer.layer.0.ff.layer_1.bias requires_grad= True
transformer.layer.0.ff.layer_2.weight requires_grad= True
transformer.layer.0.ff.layer_2.bias requires_grad= True
transformer.layer.1.rel_attn.q requires_grad= True
transformer.layer.1.rel_attn.k requires_grad= True
transformer.layer.1.rel_attn.v requires_grad= True
transformer.layer.1.rel_attn.o requires_grad= True
transformer.layer.1.rel_attn.r requires_grad= True
transformer.layer.1.rel_attn.r_r_bias requires_grad= True
transformer.layer.1.rel_attn.r_s_bias requires_grad= True
transformer.layer.1.rel_attn.r_w_bias requires_grad= True
transformer.layer.1.rel_attn.seg_embed requires_grad= True
transformer.layer.1.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.1.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.1.ff.layer_norm.weight requires_grad= True
transformer.layer.1.ff.layer_norm.bias requires_grad= True
transformer.layer.1.ff.layer_1.weight requires_grad= True
transformer.layer.1.ff.layer_1.bias requires_grad= True
transformer.layer.1.ff.layer_2.weight requires_grad= True
transformer.layer.1.ff.layer_2.bias requires_grad= True
transformer.layer.2.rel_attn.q requires_grad= True
transformer.layer.2.rel_attn.k requires_grad= True
transformer.layer.2.rel_attn.v requires_grad= True
transformer.layer.2.rel_attn.o requires_grad= True
transformer.layer.2.rel_attn.r requires_grad= True
transformer.layer.2.rel_attn.r_r_bias requires_grad= True
transformer.layer.2.rel_attn.r_s_bias requires_grad= True
transformer.layer.2.rel_attn.r_w_bias requires_grad= True
transformer.layer.2.rel_attn.seg_embed requires_grad= True
transformer.layer.2.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.2.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.2.ff.layer_norm.weight requires_grad= True
transformer.layer.2.ff.layer_norm.bias requires_grad= True
transformer.layer.2.ff.layer_1.weight requires_grad= True
transformer.layer.2.ff.layer_1.bias requires_grad= True
transformer.layer.2.ff.layer_2.weight requires_grad= True
transformer.layer.2.ff.layer_2.bias requires_grad= True
```

```
transformer.layer.3.rel_attn.q requires_grad= True
transformer.layer.3.rel_attn.k requires_grad= True
transformer.layer.3.rel_attn.v requires_grad= True
transformer.layer.3.rel_attn.o requires_grad= True
transformer.layer.3.rel_attn.r requires_grad= True
transformer.layer.3.rel_attn.r_r_bias requires_grad= True
transformer.layer.3.rel_attn.r_s_bias requires_grad= True
transformer.layer.3.rel_attn.r_w_bias requires_grad= True
transformer.layer.3.rel_attn.seg_embed requires_grad= True
transformer.layer.3.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.3.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.3.ff.layer_norm.weight requires_grad= True
transformer.layer.3.ff.layer_norm.bias requires_grad= True
transformer.layer.3.ff.layer_1.weight requires_grad= True
transformer.layer.3.ff.layer_1.bias requires_grad= True
transformer.layer.3.ff.layer_2.weight requires_grad= True
transformer.layer.3.ff.layer_2.bias requires_grad= True
transformer.layer.4.rel_attn.q requires_grad= True
transformer.layer.4.rel_attn.k requires_grad= True
transformer.layer.4.rel_attn.v requires_grad= True
transformer.layer.4.rel_attn.o requires_grad= True
transformer.layer.4.rel_attn.r requires_grad= True
transformer.layer.4.rel_attn.r_r_bias requires_grad= True
transformer.layer.4.rel_attn.r_s_bias requires_grad= True
transformer.layer.4.rel_attn.r_w_bias requires_grad= True
transformer.layer.4.rel_attn.seg_embed requires_grad= True
transformer.layer.4.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.4.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.4.ff.layer_norm.weight requires_grad= True
transformer.layer.4.ff.layer_norm.bias requires_grad= True
transformer.layer.4.ff.layer_1.weight requires_grad= True
transformer.layer.4.ff.layer_1.bias requires_grad= True
transformer.layer.4.ff.layer_2.weight requires_grad= True
transformer.layer.4.ff.layer_2.bias requires_grad= True
transformer.layer.5.rel_attn.q requires_grad= True
transformer.layer.5.rel_attn.k requires_grad= True
transformer.layer.5.rel_attn.v requires_grad= True
transformer.layer.5.rel_attn.o requires_grad= True
transformer.layer.5.rel_attn.r requires_grad= True
transformer.layer.5.rel_attn.r_r_bias requires_grad= True
transformer.layer.5.rel_attn.r_s_bias requires_grad= True
transformer.layer.5.rel_attn.r_w_bias requires_grad= True
transformer.layer.5.rel_attn.seg_embed requires_grad= True
transformer.layer.5.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.5.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.5.ff.layer_norm.weight requires_grad= True
transformer.layer.5.ff.layer_norm.bias requires_grad= True
transformer.layer.5.ff.layer_1.weight requires_grad= True
```

```
transformer.layer.5.ff.layer_1.bias requires_grad= True
transformer.layer.5.ff.layer_2.weight requires_grad= True
transformer.layer.5.ff.layer_2.bias requires_grad= True
transformer.layer.6.rel_attn.q requires_grad= True
transformer.layer.6.rel_attn.k requires_grad= True
transformer.layer.6.rel_attn.v requires_grad= True
transformer.layer.6.rel_attn.o requires_grad= True
transformer.layer.6.rel_attn.r requires_grad= True
transformer.layer.6.rel_attn.r_r_bias requires_grad= True
transformer.layer.6.rel_attn.r_s_bias requires_grad= True
transformer.layer.6.rel_attn.r_w_bias requires_grad= True
transformer.layer.6.rel_attn.seg_embed requires_grad= True
transformer.layer.6.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.6.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.6.ff.layer_norm.weight requires_grad= True
transformer.layer.6.ff.layer_norm.bias requires_grad= True
transformer.layer.6.ff.layer_1.weight requires_grad= True
transformer.layer.6.ff.layer_1.bias requires_grad= True
transformer.layer.6.ff.layer_2.weight requires_grad= True
transformer.layer.6.ff.layer_2.bias requires_grad= True
transformer.layer.7.rel_attn.q requires_grad= True
transformer.layer.7.rel_attn.k requires_grad= True
transformer.layer.7.rel_attn.v requires_grad= True
transformer.layer.7.rel_attn.o requires_grad= True
transformer.layer.7.rel_attn.r requires_grad= True
transformer.layer.7.rel_attn.r_r_bias requires_grad= True
transformer.layer.7.rel_attn.r_s_bias requires_grad= True
transformer.layer.7.rel_attn.r_w_bias requires_grad= True
transformer.layer.7.rel_attn.seg_embed requires_grad= True
transformer.layer.7.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.7.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.7.ff.layer_norm.weight requires_grad= True
transformer.layer.7.ff.layer_norm.bias requires_grad= True
transformer.layer.7.ff.layer_1.weight requires_grad= True
transformer.layer.7.ff.layer_1.bias requires_grad= True
transformer.layer.7.ff.layer_2.weight requires_grad= True
transformer.layer.7.ff.layer_2.bias requires_grad= True
transformer.layer.8.rel_attn.q requires_grad= True
transformer.layer.8.rel_attn.k requires_grad= True
transformer.layer.8.rel_attn.v requires_grad= True
transformer.layer.8.rel_attn.o requires_grad= True
transformer.layer.8.rel_attn.r requires_grad= True
transformer.layer.8.rel_attn.r_r_bias requires_grad= True
transformer.layer.8.rel_attn.r_s_bias requires_grad= True
transformer.layer.8.rel_attn.r_w_bias requires_grad= True
transformer.layer.8.rel_attn.seg_embed requires_grad= True
transformer.layer.8.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.8.rel_attn.layer_norm.bias requires_grad= True
```

```
transformer.layer.8.ff.layer_norm.weight requires_grad= True
transformer.layer.8.ff.layer_norm.bias requires_grad= True
transformer.layer.8.ff.layer_1.weight requires_grad= True
transformer.layer.8.ff.layer_1.bias requires_grad= True
transformer.layer.8.ff.layer_2.weight requires_grad= True
transformer.layer.8.ff.layer_2.bias requires_grad= True
transformer.layer.9.rel_attn.q requires_grad= True
transformer.layer.9.rel_attn.k requires_grad= True
transformer.layer.9.rel_attn.v requires_grad= True
transformer.layer.9.rel_attn.o requires_grad= True
transformer.layer.9.rel_attn.r requires_grad= True
transformer.layer.9.rel_attn.r_r_bias requires_grad= True
transformer.layer.9.rel_attn.r_s_bias requires_grad= True
transformer.layer.9.rel_attn.r_w_bias requires_grad= True
transformer.layer.9.rel_attn.seg_embed requires_grad= True
transformer.layer.9.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.9.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.9.ff.layer_norm.weight requires_grad= True
transformer.layer.9.ff.layer_norm.bias requires_grad= True
transformer.layer.9.ff.layer_1.weight requires_grad= True
transformer.layer.9.ff.layer_1.bias requires_grad= True
transformer.layer.9.ff.layer_2.weight requires_grad= True
transformer.layer.9.ff.layer_2.bias requires_grad= True
transformer.layer.10.rel_attn.q requires_grad= True
transformer.layer.10.rel_attn.k requires_grad= True
transformer.layer.10.rel_attn.v requires_grad= True
transformer.layer.10.rel_attn.o requires_grad= True
transformer.layer.10.rel_attn.r requires_grad= True
transformer.layer.10.rel_attn.r_r_bias requires_grad= True
transformer.layer.10.rel_attn.r_s_bias requires_grad= True
transformer.layer.10.rel_attn.r_w_bias requires_grad= True
transformer.layer.10.rel_attn.seg_embed requires_grad= True
transformer.layer.10.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.10.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.10.ff.layer_norm.weight requires_grad= True
transformer.layer.10.ff.layer_norm.bias requires_grad= True
transformer.layer.10.ff.layer_1.weight requires_grad= True
transformer.layer.10.ff.layer_1.bias requires_grad= True
transformer.layer.10.ff.layer_2.weight requires_grad= True
transformer.layer.10.ff.layer_2.bias requires_grad= True
transformer.layer.11.rel_attn.q requires_grad= True
transformer.layer.11.rel_attn.k requires_grad= True
transformer.layer.11.rel_attn.v requires_grad= True
transformer.layer.11.rel_attn.o requires_grad= True
transformer.layer.11.rel_attn.r requires_grad= True
transformer.layer.11.rel_attn.r_r_bias requires_grad= True
transformer.layer.11.rel_attn.r_s_bias requires_grad= True
transformer.layer.11.rel_attn.r_w_bias requires_grad= True
```

```
transformer.layer.11.rel_attn.seg_embed requires_grad= True
transformer.layer.11.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.11.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.11.ff.layer_norm.weight requires_grad= True
transformer.layer.11.ff.layer_norm.bias requires_grad= True
transformer.layer.11.ff.layer_1.weight requires_grad= True
transformer.layer.11.ff.layer_1.bias requires_grad= True
transformer.layer.11.ff.layer_2.weight requires_grad= True
transformer.layer.11.ff.layer_2.bias requires_grad= True
transformer.layer.12.rel_attn.q requires_grad= True
transformer.layer.12.rel_attn.k requires_grad= True
transformer.layer.12.rel_attn.v requires_grad= True
transformer.layer.12.rel_attn.o requires_grad= True
transformer.layer.12.rel_attn.r requires_grad= True
transformer.layer.12.rel_attn.r_r_bias requires_grad= True
transformer.layer.12.rel_attn.r_s_bias requires_grad= True
transformer.layer.12.rel_attn.r_w_bias requires_grad= True
transformer.layer.12.rel_attn.seg_embed requires_grad= True
transformer.layer.12.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.12.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.12.ff.layer_norm.weight requires_grad= True
transformer.layer.12.ff.layer_norm.bias requires_grad= True
transformer.layer.12.ff.layer_1.weight requires_grad= True
transformer.layer.12.ff.layer_1.bias requires_grad= True
transformer.layer.12.ff.layer_2.weight requires_grad= True
transformer.layer.12.ff.layer_2.bias requires_grad= True
transformer.layer.13.rel_attn.q requires_grad= True
transformer.layer.13.rel_attn.k requires_grad= True
transformer.layer.13.rel_attn.v requires_grad= True
transformer.layer.13.rel_attn.o requires_grad= True
transformer.layer.13.rel_attn.r requires_grad= True
transformer.layer.13.rel_attn.r_r_bias requires_grad= True
transformer.layer.13.rel_attn.r_s_bias requires_grad= True
transformer.layer.13.rel_attn.r_w_bias requires_grad= True
transformer.layer.13.rel_attn.seg_embed requires_grad= True
transformer.layer.13.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.13.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.13.ff.layer_norm.weight requires_grad= True
transformer.layer.13.ff.layer_norm.bias requires_grad= True
transformer.layer.13.ff.layer_1.weight requires_grad= True
transformer.layer.13.ff.layer_1.bias requires_grad= True
transformer.layer.13.ff.layer_2.weight requires_grad= True
transformer.layer.13.ff.layer_2.bias requires_grad= True
transformer.layer.14.rel_attn.q requires_grad= True
transformer.layer.14.rel_attn.k requires_grad= True
transformer.layer.14.rel_attn.v requires_grad= True
transformer.layer.14.rel_attn.o requires_grad= True
transformer.layer.14.rel_attn.r requires_grad= True
```

```
transformer.layer.14.rel_attn.r_r_bias requires_grad= True
transformer.layer.14.rel_attn.r_s_bias requires_grad= True
transformer.layer.14.rel_attn.r_w_bias requires_grad= True
transformer.layer.14.rel_attn.seg_embed requires_grad= True
transformer.layer.14.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.14.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.14.ff.layer_norm.weight requires_grad= True
transformer.layer.14.ff.layer_norm.bias requires_grad= True
transformer.layer.14.ff.layer_1.weight requires_grad= True
transformer.layer.14.ff.layer_1.bias requires_grad= True
transformer.layer.14.ff.layer_2.weight requires_grad= True
transformer.layer.14.ff.layer_2.bias requires_grad= True
transformer.layer.15.rel_attn.q requires_grad= True
transformer.layer.15.rel_attn.k requires_grad= True
transformer.layer.15.rel_attn.v requires_grad= True
transformer.layer.15.rel_attn.o requires_grad= True
transformer.layer.15.rel_attn.r requires_grad= True
transformer.layer.15.rel_attn.r_r_bias requires_grad= True
transformer.layer.15.rel_attn.r_s_bias requires_grad= True
transformer.layer.15.rel_attn.r_w_bias requires_grad= True
transformer.layer.15.rel_attn.seg_embed requires_grad= True
transformer.layer.15.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.15.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.15.ff.layer_norm.weight requires_grad= True
transformer.layer.15.ff.layer_norm.bias requires_grad= True
transformer.layer.15.ff.layer_1.weight requires_grad= True
transformer.layer.15.ff.layer_1.bias requires_grad= True
transformer.layer.15.ff.layer_2.weight requires_grad= True
transformer.layer.15.ff.layer_2.bias requires_grad= True
transformer.layer.16.rel_attn.q requires_grad= True
transformer.layer.16.rel_attn.k requires_grad= True
transformer.layer.16.rel_attn.v requires_grad= True
transformer.layer.16.rel_attn.o requires_grad= True
transformer.layer.16.rel_attn.r requires_grad= True
transformer.layer.16.rel_attn.r_r_bias requires_grad= True
transformer.layer.16.rel_attn.r_s_bias requires_grad= True
transformer.layer.16.rel_attn.r_w_bias requires_grad= True
transformer.layer.16.rel_attn.seg_embed requires_grad= True
transformer.layer.16.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.16.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.16.ff.layer_norm.weight requires_grad= True
transformer.layer.16.ff.layer_norm.bias requires_grad= True
transformer.layer.16.ff.layer_1.weight requires_grad= True
transformer.layer.16.ff.layer_1.bias requires_grad= True
transformer.layer.16.ff.layer_2.weight requires_grad= True
transformer.layer.16.ff.layer_2.bias requires_grad= True
transformer.layer.17.rel_attn.q requires_grad= True
transformer.layer.17.rel_attn.k requires_grad= True
```

```
transformer.layer.17.rel_attn.v requires_grad= True
transformer.layer.17.rel_attn.o requires_grad= True
transformer.layer.17.rel_attn.r requires_grad= True
transformer.layer.17.rel_attn.r_r_bias requires_grad= True
transformer.layer.17.rel_attn.r_s_bias requires_grad= True
transformer.layer.17.rel_attn.r_w_bias requires_grad= True
transformer.layer.17.rel_attn.seg_embed requires_grad= True
transformer.layer.17.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.17.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.17.ff.layer_norm.weight requires_grad= True
transformer.layer.17.ff.layer_norm.bias requires_grad= True
transformer.layer.17.ff.layer_1.weight requires_grad= True
transformer.layer.17.ff.layer_1.bias requires_grad= True
transformer.layer.17.ff.layer_2.weight requires_grad= True
transformer.layer.17.ff.layer_2.bias requires_grad= True
transformer.layer.18.rel_attn.q requires_grad= True
transformer.layer.18.rel_attn.k requires_grad= True
transformer.layer.18.rel_attn.v requires_grad= True
transformer.layer.18.rel_attn.o requires_grad= True
transformer.layer.18.rel_attn.r requires_grad= True
transformer.layer.18.rel_attn.r_r_bias requires_grad= True
transformer.layer.18.rel_attn.r_s_bias requires_grad= True
transformer.layer.18.rel_attn.r_w_bias requires_grad= True
transformer.layer.18.rel_attn.seg_embed requires_grad= True
transformer.layer.18.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.18.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.18.ff.layer_norm.weight requires_grad= True
transformer.layer.18.ff.layer_norm.bias requires_grad= True
transformer.layer.18.ff.layer_1.weight requires_grad= True
transformer.layer.18.ff.layer_1.bias requires_grad= True
transformer.layer.18.ff.layer_2.weight requires_grad= True
transformer.layer.18.ff.layer_2.bias requires_grad= True
transformer.layer.19.rel_attn.q requires_grad= True
transformer.layer.19.rel_attn.k requires_grad= True
transformer.layer.19.rel_attn.v requires_grad= True
transformer.layer.19.rel_attn.o requires_grad= True
transformer.layer.19.rel_attn.r requires_grad= True
transformer.layer.19.rel_attn.r_r_bias requires_grad= True
transformer.layer.19.rel_attn.r_s_bias requires_grad= True
transformer.layer.19.rel_attn.r_w_bias requires_grad= True
transformer.layer.19.rel_attn.seg_embed requires_grad= True
transformer.layer.19.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.19.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.19.ff.layer_norm.weight requires_grad= True
transformer.layer.19.ff.layer_norm.bias requires_grad= True
transformer.layer.19.ff.layer_1.weight requires_grad= True
transformer.layer.19.ff.layer_1.bias requires_grad= True
transformer.layer.19.ff.layer_2.weight requires_grad= True
```

```
transformer.layer.19.ff.layer_2.bias requires_grad= True
transformer.layer.20.rel_attn.q requires_grad= True
transformer.layer.20.rel_attn.k requires_grad= True
transformer.layer.20.rel_attn.v requires_grad= True
transformer.layer.20.rel_attn.o requires_grad= True
transformer.layer.20.rel_attn.r requires_grad= True
transformer.layer.20.rel_attn.r_r_bias requires_grad= True
transformer.layer.20.rel_attn.r_s_bias requires_grad= True
transformer.layer.20.rel_attn.r_w_bias requires_grad= True
transformer.layer.20.rel_attn.seg_embed requires_grad= True
transformer.layer.20.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.20.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.20.ff.layer_norm.weight requires_grad= True
transformer.layer.20.ff.layer_norm.bias requires_grad= True
transformer.layer.20.ff.layer_1.weight requires_grad= True
transformer.layer.20.ff.layer_1.bias requires_grad= True
transformer.layer.20.ff.layer_2.weight requires_grad= True
transformer.layer.20.ff.layer_2.bias requires_grad= True
transformer.layer.21.rel_attn.q requires_grad= True
transformer.layer.21.rel_attn.k requires_grad= True
transformer.layer.21.rel_attn.v requires_grad= True
transformer.layer.21.rel_attn.o requires_grad= True
transformer.layer.21.rel_attn.r requires_grad= True
transformer.layer.21.rel_attn.r_r_bias requires_grad= True
transformer.layer.21.rel_attn.r_s_bias requires_grad= True
transformer.layer.21.rel_attn.r_w_bias requires_grad= True
transformer.layer.21.rel_attn.seg_embed requires_grad= True
transformer.layer.21.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.21.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.21.ff.layer_norm.weight requires_grad= True
transformer.layer.21.ff.layer_norm.bias requires_grad= True
transformer.layer.21.ff.layer_1.weight requires_grad= True
transformer.layer.21.ff.layer_1.bias requires_grad= True
transformer.layer.21.ff.layer_2.weight requires_grad= True
transformer.layer.21.ff.layer_2.bias requires_grad= True
transformer.layer.22.rel_attn.q requires_grad= True
transformer.layer.22.rel_attn.k requires_grad= True
transformer.layer.22.rel_attn.v requires_grad= True
transformer.layer.22.rel_attn.o requires_grad= True
transformer.layer.22.rel_attn.r requires_grad= True
transformer.layer.22.rel_attn.r_r_bias requires_grad= True
transformer.layer.22.rel_attn.r_s_bias requires_grad= True
transformer.layer.22.rel_attn.r_w_bias requires_grad= True
transformer.layer.22.rel_attn.seg_embed requires_grad= True
transformer.layer.22.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.22.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.22.ff.layer_norm.weight requires_grad= True
transformer.layer.22.ff.layer_norm.bias requires_grad= True
```

```
transformer.layer.22.ff.layer_1.weight requires_grad= True
transformer.layer.22.ff.layer_1.bias requires_grad= True
transformer.layer.22.ff.layer_2.weight requires_grad= True
transformer.layer.22.ff.layer_2.bias requires_grad= True
transformer.layer.23.rel_attn.q requires_grad= True
transformer.layer.23.rel_attn.k requires_grad= True
transformer.layer.23.rel_attn.v requires_grad= True
transformer.layer.23.rel_attn.o requires_grad= True
transformer.layer.23.rel_attn.r requires_grad= True
transformer.layer.23.rel_attn.r_r_bias requires_grad= True
transformer.layer.23.rel_attn.r_s_bias requires_grad= True
transformer.layer.23.rel_attn.r_w_bias requires_grad= True
transformer.layer.23.rel_attn.seg_embed requires_grad= True
transformer.layer.23.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.23.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.23.ff.layer_norm.weight requires_grad= True
transformer.layer.23.ff.layer_norm.bias requires_grad= True
transformer.layer.23.ff.layer_1.weight requires_grad= True
transformer.layer.23.ff.layer_1.bias requires_grad= True
transformer.layer.23.ff.layer_2.weight requires_grad= True
transformer.layer.23.ff.layer_2.bias requires_grad= True
sequence_summary.summary.weight requires_grad= True
sequence_summary.summary.bias requires_grad= True
logits_proj.weight requires_grad= True
logits_proj.bias requires_grad= True
```

[104]:
```
##########################################
layers_to_unfreeze = [
"transformer.layer.23.rel_attn.q",
"transformer.layer.23.rel_attn.k",
"transformer.layer.23.rel_attn.v",
"transformer.layer.23.rel_attn.o",
"transformer.layer.23.rel_attn.r",
"transformer.layer.23.rel_attn.r_r_bias",
"transformer.layer.23.rel_attn.r_s_bias",
"transformer.layer.23.rel_attn.r_w_bias",
"transformer.layer.23.rel_attn.seg_embed",
"transformer.layer.23.rel_attn.layer_norm.weight",
"transformer.layer.23.rel_attn.layer_norm.bias",
"transformer.layer.23.ff.layer_norm.weight",
"transformer.layer.23.ff.layer_norm.bias",
"transformer.layer.23.ff.layer_1.weight",
"transformer.layer.23.ff.layer_1.bias",
"transformer.layer.23.ff.layer_2.weight",
"transformer.layer.23.ff.layer_2.bias",
"sequence_summary.summary.weight",
"sequence_summary.summary.bias",
```

```
"logits_proj.weight",
"logits_proj.bias"
    ]
freeze_unfreeze_layers(model, layers_to_unfreeze=layers_to_unfreeze)
print(model.config)
print("=============")
print("num_parameters:", model.num_parameters())
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
print("=============")
########################################
print("Experiment configuration used with this experiment:")
print("model used:", named_model)
print("learning rate used:", learning_rate)
print("number of epochs:", num_epochs)
print("maximum sequence length:", length_max)
print("batch size used:", size_batch)
print("regularization value:", regularization_weight_decay)
print("outcome variable:", y_col)
print("task:", x_task)
print("input column:", x_col)
########################################
print("=============")
print("num_trainable_parameters:", model.num_parameters(only_trainable=True))
```

```
XLNetConfig {
  "_attn_implementation_autoset": true,
  "architectures": [
    "XLNetLMHeadModel"
  ],
  "attn_type": "bi",
  "bi_data": false,
  "bos_token_id": 1,
  "clamp_len": -1,
  "d_head": 64,
  "d_inner": 4096,
  "d_model": 1024,
  "dropout": 0.1,
  "end_n_top": 5,
  "eos_token_id": 2,
  "ff_activation": "gelu",
  "initializer_range": 0.02,
  "layer_norm_eps": 1e-12,
  "mem_len": null,
  "model_type": "xlnet",
  "n_head": 16,
  "n_layer": 24,
  "pad_token_id": 5,
  "reuse_len": null,
```

```
  "same_length": false,
  "start_n_top": 5,
  "summary_activation": "tanh",
  "summary_last_dropout": 0.1,
  "summary_type": "last",
  "summary_use_proj": true,
  "task_specific_params": {
    "text-generation": {
      "do_sample": true,
      "max_length": 250
    }
  },
  "torch_dtype": "float32",
  "transformers_version": "4.50.3",
  "untie_r": true,
  "use_mems_eval": true,
  "use_mems_train": false,
  "vocab_size": 32000
}

=============
num_parameters: 361320450
num_trainable_parameters: 14697474
=============
Experiment configuration used with this experiment:
model used: xlnet/xlnet-large-cased
learning rate used: 5e-06
number of epochs: 1
maximum sequence length: 128
batch size used: 128
regularization value: 0.5
outcome variable: binary_complexity
task: single
input column: sentence_no_contractions
=============
num_trainable_parameters: 14697474
```

[105]:
```
for name, param in model.named_parameters():
    print(name, "requires_grad=", param.requires_grad)
```

```
transformer.mask_emb requires_grad= False
transformer.word_embedding.weight requires_grad= False
transformer.layer.0.rel_attn.q requires_grad= False
transformer.layer.0.rel_attn.k requires_grad= False
transformer.layer.0.rel_attn.v requires_grad= False
transformer.layer.0.rel_attn.o requires_grad= False
transformer.layer.0.rel_attn.r requires_grad= False
transformer.layer.0.rel_attn.r_r_bias requires_grad= False
```

```
transformer.layer.0.rel_attn.r_s_bias requires_grad= False
transformer.layer.0.rel_attn.r_w_bias requires_grad= False
transformer.layer.0.rel_attn.seg_embed requires_grad= False
transformer.layer.0.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.0.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.0.ff.layer_norm.weight requires_grad= False
transformer.layer.0.ff.layer_norm.bias requires_grad= False
transformer.layer.0.ff.layer_1.weight requires_grad= False
transformer.layer.0.ff.layer_1.bias requires_grad= False
transformer.layer.0.ff.layer_2.weight requires_grad= False
transformer.layer.0.ff.layer_2.bias requires_grad= False
transformer.layer.1.rel_attn.q requires_grad= False
transformer.layer.1.rel_attn.k requires_grad= False
transformer.layer.1.rel_attn.v requires_grad= False
transformer.layer.1.rel_attn.o requires_grad= False
transformer.layer.1.rel_attn.r requires_grad= False
transformer.layer.1.rel_attn.r_r_bias requires_grad= False
transformer.layer.1.rel_attn.r_s_bias requires_grad= False
transformer.layer.1.rel_attn.r_w_bias requires_grad= False
transformer.layer.1.rel_attn.seg_embed requires_grad= False
transformer.layer.1.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.1.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.1.ff.layer_norm.weight requires_grad= False
transformer.layer.1.ff.layer_norm.bias requires_grad= False
transformer.layer.1.ff.layer_1.weight requires_grad= False
transformer.layer.1.ff.layer_1.bias requires_grad= False
transformer.layer.1.ff.layer_2.weight requires_grad= False
transformer.layer.1.ff.layer_2.bias requires_grad= False
transformer.layer.2.rel_attn.q requires_grad= False
transformer.layer.2.rel_attn.k requires_grad= False
transformer.layer.2.rel_attn.v requires_grad= False
transformer.layer.2.rel_attn.o requires_grad= False
transformer.layer.2.rel_attn.r requires_grad= False
transformer.layer.2.rel_attn.r_r_bias requires_grad= False
transformer.layer.2.rel_attn.r_s_bias requires_grad= False
transformer.layer.2.rel_attn.r_w_bias requires_grad= False
transformer.layer.2.rel_attn.seg_embed requires_grad= False
transformer.layer.2.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.2.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.2.ff.layer_norm.weight requires_grad= False
transformer.layer.2.ff.layer_norm.bias requires_grad= False
transformer.layer.2.ff.layer_1.weight requires_grad= False
transformer.layer.2.ff.layer_1.bias requires_grad= False
transformer.layer.2.ff.layer_2.weight requires_grad= False
transformer.layer.2.ff.layer_2.bias requires_grad= False
transformer.layer.3.rel_attn.q requires_grad= False
transformer.layer.3.rel_attn.k requires_grad= False
transformer.layer.3.rel_attn.v requires_grad= False
```

```
transformer.layer.3.rel_attn.o requires_grad= False
transformer.layer.3.rel_attn.r requires_grad= False
transformer.layer.3.rel_attn.r_r_bias requires_grad= False
transformer.layer.3.rel_attn.r_s_bias requires_grad= False
transformer.layer.3.rel_attn.r_w_bias requires_grad= False
transformer.layer.3.rel_attn.seg_embed requires_grad= False
transformer.layer.3.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.3.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.3.ff.layer_norm.weight requires_grad= False
transformer.layer.3.ff.layer_norm.bias requires_grad= False
transformer.layer.3.ff.layer_1.weight requires_grad= False
transformer.layer.3.ff.layer_1.bias requires_grad= False
transformer.layer.3.ff.layer_2.weight requires_grad= False
transformer.layer.3.ff.layer_2.bias requires_grad= False
transformer.layer.4.rel_attn.q requires_grad= False
transformer.layer.4.rel_attn.k requires_grad= False
transformer.layer.4.rel_attn.v requires_grad= False
transformer.layer.4.rel_attn.o requires_grad= False
transformer.layer.4.rel_attn.r requires_grad= False
transformer.layer.4.rel_attn.r_r_bias requires_grad= False
transformer.layer.4.rel_attn.r_s_bias requires_grad= False
transformer.layer.4.rel_attn.r_w_bias requires_grad= False
transformer.layer.4.rel_attn.seg_embed requires_grad= False
transformer.layer.4.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.4.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.4.ff.layer_norm.weight requires_grad= False
transformer.layer.4.ff.layer_norm.bias requires_grad= False
transformer.layer.4.ff.layer_1.weight requires_grad= False
transformer.layer.4.ff.layer_1.bias requires_grad= False
transformer.layer.4.ff.layer_2.weight requires_grad= False
transformer.layer.4.ff.layer_2.bias requires_grad= False
transformer.layer.5.rel_attn.q requires_grad= False
transformer.layer.5.rel_attn.k requires_grad= False
transformer.layer.5.rel_attn.v requires_grad= False
transformer.layer.5.rel_attn.o requires_grad= False
transformer.layer.5.rel_attn.r requires_grad= False
transformer.layer.5.rel_attn.r_r_bias requires_grad= False
transformer.layer.5.rel_attn.r_s_bias requires_grad= False
transformer.layer.5.rel_attn.r_w_bias requires_grad= False
transformer.layer.5.rel_attn.seg_embed requires_grad= False
transformer.layer.5.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.5.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.5.ff.layer_norm.weight requires_grad= False
transformer.layer.5.ff.layer_norm.bias requires_grad= False
transformer.layer.5.ff.layer_1.weight requires_grad= False
transformer.layer.5.ff.layer_1.bias requires_grad= False
transformer.layer.5.ff.layer_2.weight requires_grad= False
transformer.layer.5.ff.layer_2.bias requires_grad= False
```

```
transformer.layer.6.rel_attn.q requires_grad= False
transformer.layer.6.rel_attn.k requires_grad= False
transformer.layer.6.rel_attn.v requires_grad= False
transformer.layer.6.rel_attn.o requires_grad= False
transformer.layer.6.rel_attn.r requires_grad= False
transformer.layer.6.rel_attn.r_r_bias requires_grad= False
transformer.layer.6.rel_attn.r_s_bias requires_grad= False
transformer.layer.6.rel_attn.r_w_bias requires_grad= False
transformer.layer.6.rel_attn.seg_embed requires_grad= False
transformer.layer.6.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.6.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.6.ff.layer_norm.weight requires_grad= False
transformer.layer.6.ff.layer_norm.bias requires_grad= False
transformer.layer.6.ff.layer_1.weight requires_grad= False
transformer.layer.6.ff.layer_1.bias requires_grad= False
transformer.layer.6.ff.layer_2.weight requires_grad= False
transformer.layer.6.ff.layer_2.bias requires_grad= False
transformer.layer.7.rel_attn.q requires_grad= False
transformer.layer.7.rel_attn.k requires_grad= False
transformer.layer.7.rel_attn.v requires_grad= False
transformer.layer.7.rel_attn.o requires_grad= False
transformer.layer.7.rel_attn.r requires_grad= False
transformer.layer.7.rel_attn.r_r_bias requires_grad= False
transformer.layer.7.rel_attn.r_s_bias requires_grad= False
transformer.layer.7.rel_attn.r_w_bias requires_grad= False
transformer.layer.7.rel_attn.seg_embed requires_grad= False
transformer.layer.7.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.7.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.7.ff.layer_norm.weight requires_grad= False
transformer.layer.7.ff.layer_norm.bias requires_grad= False
transformer.layer.7.ff.layer_1.weight requires_grad= False
transformer.layer.7.ff.layer_1.bias requires_grad= False
transformer.layer.7.ff.layer_2.weight requires_grad= False
transformer.layer.7.ff.layer_2.bias requires_grad= False
transformer.layer.8.rel_attn.q requires_grad= False
transformer.layer.8.rel_attn.k requires_grad= False
transformer.layer.8.rel_attn.v requires_grad= False
transformer.layer.8.rel_attn.o requires_grad= False
transformer.layer.8.rel_attn.r requires_grad= False
transformer.layer.8.rel_attn.r_r_bias requires_grad= False
transformer.layer.8.rel_attn.r_s_bias requires_grad= False
transformer.layer.8.rel_attn.r_w_bias requires_grad= False
transformer.layer.8.rel_attn.seg_embed requires_grad= False
transformer.layer.8.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.8.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.8.ff.layer_norm.weight requires_grad= False
transformer.layer.8.ff.layer_norm.bias requires_grad= False
transformer.layer.8.ff.layer_1.weight requires_grad= False
```

```
transformer.layer.8.ff.layer_1.bias requires_grad= False
transformer.layer.8.ff.layer_2.weight requires_grad= False
transformer.layer.8.ff.layer_2.bias requires_grad= False
transformer.layer.9.rel_attn.q requires_grad= False
transformer.layer.9.rel_attn.k requires_grad= False
transformer.layer.9.rel_attn.v requires_grad= False
transformer.layer.9.rel_attn.o requires_grad= False
transformer.layer.9.rel_attn.r requires_grad= False
transformer.layer.9.rel_attn.r_r_bias requires_grad= False
transformer.layer.9.rel_attn.r_s_bias requires_grad= False
transformer.layer.9.rel_attn.r_w_bias requires_grad= False
transformer.layer.9.rel_attn.seg_embed requires_grad= False
transformer.layer.9.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.9.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.9.ff.layer_norm.weight requires_grad= False
transformer.layer.9.ff.layer_norm.bias requires_grad= False
transformer.layer.9.ff.layer_1.weight requires_grad= False
transformer.layer.9.ff.layer_1.bias requires_grad= False
transformer.layer.9.ff.layer_2.weight requires_grad= False
transformer.layer.9.ff.layer_2.bias requires_grad= False
transformer.layer.10.rel_attn.q requires_grad= False
transformer.layer.10.rel_attn.k requires_grad= False
transformer.layer.10.rel_attn.v requires_grad= False
transformer.layer.10.rel_attn.o requires_grad= False
transformer.layer.10.rel_attn.r requires_grad= False
transformer.layer.10.rel_attn.r_r_bias requires_grad= False
transformer.layer.10.rel_attn.r_s_bias requires_grad= False
transformer.layer.10.rel_attn.r_w_bias requires_grad= False
transformer.layer.10.rel_attn.seg_embed requires_grad= False
transformer.layer.10.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.10.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.10.ff.layer_norm.weight requires_grad= False
transformer.layer.10.ff.layer_norm.bias requires_grad= False
transformer.layer.10.ff.layer_1.weight requires_grad= False
transformer.layer.10.ff.layer_1.bias requires_grad= False
transformer.layer.10.ff.layer_2.weight requires_grad= False
transformer.layer.10.ff.layer_2.bias requires_grad= False
transformer.layer.11.rel_attn.q requires_grad= False
transformer.layer.11.rel_attn.k requires_grad= False
transformer.layer.11.rel_attn.v requires_grad= False
transformer.layer.11.rel_attn.o requires_grad= False
transformer.layer.11.rel_attn.r requires_grad= False
transformer.layer.11.rel_attn.r_r_bias requires_grad= False
transformer.layer.11.rel_attn.r_s_bias requires_grad= False
transformer.layer.11.rel_attn.r_w_bias requires_grad= False
transformer.layer.11.rel_attn.seg_embed requires_grad= False
transformer.layer.11.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.11.rel_attn.layer_norm.bias requires_grad= False
```

```
transformer.layer.11.ff.layer_norm.weight requires_grad= False
transformer.layer.11.ff.layer_norm.bias requires_grad= False
transformer.layer.11.ff.layer_1.weight requires_grad= False
transformer.layer.11.ff.layer_1.bias requires_grad= False
transformer.layer.11.ff.layer_2.weight requires_grad= False
transformer.layer.11.ff.layer_2.bias requires_grad= False
transformer.layer.12.rel_attn.q requires_grad= False
transformer.layer.12.rel_attn.k requires_grad= False
transformer.layer.12.rel_attn.v requires_grad= False
transformer.layer.12.rel_attn.o requires_grad= False
transformer.layer.12.rel_attn.r requires_grad= False
transformer.layer.12.rel_attn.r_r_bias requires_grad= False
transformer.layer.12.rel_attn.r_s_bias requires_grad= False
transformer.layer.12.rel_attn.r_w_bias requires_grad= False
transformer.layer.12.rel_attn.seg_embed requires_grad= False
transformer.layer.12.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.12.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.12.ff.layer_norm.weight requires_grad= False
transformer.layer.12.ff.layer_norm.bias requires_grad= False
transformer.layer.12.ff.layer_1.weight requires_grad= False
transformer.layer.12.ff.layer_1.bias requires_grad= False
transformer.layer.12.ff.layer_2.weight requires_grad= False
transformer.layer.12.ff.layer_2.bias requires_grad= False
transformer.layer.13.rel_attn.q requires_grad= False
transformer.layer.13.rel_attn.k requires_grad= False
transformer.layer.13.rel_attn.v requires_grad= False
transformer.layer.13.rel_attn.o requires_grad= False
transformer.layer.13.rel_attn.r requires_grad= False
transformer.layer.13.rel_attn.r_r_bias requires_grad= False
transformer.layer.13.rel_attn.r_s_bias requires_grad= False
transformer.layer.13.rel_attn.r_w_bias requires_grad= False
transformer.layer.13.rel_attn.seg_embed requires_grad= False
transformer.layer.13.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.13.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.13.ff.layer_norm.weight requires_grad= False
transformer.layer.13.ff.layer_norm.bias requires_grad= False
transformer.layer.13.ff.layer_1.weight requires_grad= False
transformer.layer.13.ff.layer_1.bias requires_grad= False
transformer.layer.13.ff.layer_2.weight requires_grad= False
transformer.layer.13.ff.layer_2.bias requires_grad= False
transformer.layer.14.rel_attn.q requires_grad= False
transformer.layer.14.rel_attn.k requires_grad= False
transformer.layer.14.rel_attn.v requires_grad= False
transformer.layer.14.rel_attn.o requires_grad= False
transformer.layer.14.rel_attn.r requires_grad= False
transformer.layer.14.rel_attn.r_r_bias requires_grad= False
transformer.layer.14.rel_attn.r_s_bias requires_grad= False
transformer.layer.14.rel_attn.r_w_bias requires_grad= False
```

```
transformer.layer.14.rel_attn.seg_embed requires_grad= False
transformer.layer.14.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.14.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.14.ff.layer_norm.weight requires_grad= False
transformer.layer.14.ff.layer_norm.bias requires_grad= False
transformer.layer.14.ff.layer_1.weight requires_grad= False
transformer.layer.14.ff.layer_1.bias requires_grad= False
transformer.layer.14.ff.layer_2.weight requires_grad= False
transformer.layer.14.ff.layer_2.bias requires_grad= False
transformer.layer.15.rel_attn.q requires_grad= False
transformer.layer.15.rel_attn.k requires_grad= False
transformer.layer.15.rel_attn.v requires_grad= False
transformer.layer.15.rel_attn.o requires_grad= False
transformer.layer.15.rel_attn.r requires_grad= False
transformer.layer.15.rel_attn.r_r_bias requires_grad= False
transformer.layer.15.rel_attn.r_s_bias requires_grad= False
transformer.layer.15.rel_attn.r_w_bias requires_grad= False
transformer.layer.15.rel_attn.seg_embed requires_grad= False
transformer.layer.15.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.15.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.15.ff.layer_norm.weight requires_grad= False
transformer.layer.15.ff.layer_norm.bias requires_grad= False
transformer.layer.15.ff.layer_1.weight requires_grad= False
transformer.layer.15.ff.layer_1.bias requires_grad= False
transformer.layer.15.ff.layer_2.weight requires_grad= False
transformer.layer.15.ff.layer_2.bias requires_grad= False
transformer.layer.16.rel_attn.q requires_grad= False
transformer.layer.16.rel_attn.k requires_grad= False
transformer.layer.16.rel_attn.v requires_grad= False
transformer.layer.16.rel_attn.o requires_grad= False
transformer.layer.16.rel_attn.r requires_grad= False
transformer.layer.16.rel_attn.r_r_bias requires_grad= False
transformer.layer.16.rel_attn.r_s_bias requires_grad= False
transformer.layer.16.rel_attn.r_w_bias requires_grad= False
transformer.layer.16.rel_attn.seg_embed requires_grad= False
transformer.layer.16.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.16.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.16.ff.layer_norm.weight requires_grad= False
transformer.layer.16.ff.layer_norm.bias requires_grad= False
transformer.layer.16.ff.layer_1.weight requires_grad= False
transformer.layer.16.ff.layer_1.bias requires_grad= False
transformer.layer.16.ff.layer_2.weight requires_grad= False
transformer.layer.16.ff.layer_2.bias requires_grad= False
transformer.layer.17.rel_attn.q requires_grad= False
transformer.layer.17.rel_attn.k requires_grad= False
transformer.layer.17.rel_attn.v requires_grad= False
transformer.layer.17.rel_attn.o requires_grad= False
transformer.layer.17.rel_attn.r requires_grad= False
```

```
transformer.layer.17.rel_attn.r_r_bias requires_grad= False
transformer.layer.17.rel_attn.r_s_bias requires_grad= False
transformer.layer.17.rel_attn.r_w_bias requires_grad= False
transformer.layer.17.rel_attn.seg_embed requires_grad= False
transformer.layer.17.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.17.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.17.ff.layer_norm.weight requires_grad= False
transformer.layer.17.ff.layer_norm.bias requires_grad= False
transformer.layer.17.ff.layer_1.weight requires_grad= False
transformer.layer.17.ff.layer_1.bias requires_grad= False
transformer.layer.17.ff.layer_2.weight requires_grad= False
transformer.layer.17.ff.layer_2.bias requires_grad= False
transformer.layer.18.rel_attn.q requires_grad= False
transformer.layer.18.rel_attn.k requires_grad= False
transformer.layer.18.rel_attn.v requires_grad= False
transformer.layer.18.rel_attn.o requires_grad= False
transformer.layer.18.rel_attn.r requires_grad= False
transformer.layer.18.rel_attn.r_r_bias requires_grad= False
transformer.layer.18.rel_attn.r_s_bias requires_grad= False
transformer.layer.18.rel_attn.r_w_bias requires_grad= False
transformer.layer.18.rel_attn.seg_embed requires_grad= False
transformer.layer.18.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.18.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.18.ff.layer_norm.weight requires_grad= False
transformer.layer.18.ff.layer_norm.bias requires_grad= False
transformer.layer.18.ff.layer_1.weight requires_grad= False
transformer.layer.18.ff.layer_1.bias requires_grad= False
transformer.layer.18.ff.layer_2.weight requires_grad= False
transformer.layer.18.ff.layer_2.bias requires_grad= False
transformer.layer.19.rel_attn.q requires_grad= False
transformer.layer.19.rel_attn.k requires_grad= False
transformer.layer.19.rel_attn.v requires_grad= False
transformer.layer.19.rel_attn.o requires_grad= False
transformer.layer.19.rel_attn.r requires_grad= False
transformer.layer.19.rel_attn.r_r_bias requires_grad= False
transformer.layer.19.rel_attn.r_s_bias requires_grad= False
transformer.layer.19.rel_attn.r_w_bias requires_grad= False
transformer.layer.19.rel_attn.seg_embed requires_grad= False
transformer.layer.19.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.19.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.19.ff.layer_norm.weight requires_grad= False
transformer.layer.19.ff.layer_norm.bias requires_grad= False
transformer.layer.19.ff.layer_1.weight requires_grad= False
transformer.layer.19.ff.layer_1.bias requires_grad= False
transformer.layer.19.ff.layer_2.weight requires_grad= False
transformer.layer.19.ff.layer_2.bias requires_grad= False
transformer.layer.20.rel_attn.q requires_grad= False
transformer.layer.20.rel_attn.k requires_grad= False
```

```
transformer.layer.20.rel_attn.v requires_grad= False
transformer.layer.20.rel_attn.o requires_grad= False
transformer.layer.20.rel_attn.r requires_grad= False
transformer.layer.20.rel_attn.r_r_bias requires_grad= False
transformer.layer.20.rel_attn.r_s_bias requires_grad= False
transformer.layer.20.rel_attn.r_w_bias requires_grad= False
transformer.layer.20.rel_attn.seg_embed requires_grad= False
transformer.layer.20.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.20.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.20.ff.layer_norm.weight requires_grad= False
transformer.layer.20.ff.layer_norm.bias requires_grad= False
transformer.layer.20.ff.layer_1.weight requires_grad= False
transformer.layer.20.ff.layer_1.bias requires_grad= False
transformer.layer.20.ff.layer_2.weight requires_grad= False
transformer.layer.20.ff.layer_2.bias requires_grad= False
transformer.layer.21.rel_attn.q requires_grad= False
transformer.layer.21.rel_attn.k requires_grad= False
transformer.layer.21.rel_attn.v requires_grad= False
transformer.layer.21.rel_attn.o requires_grad= False
transformer.layer.21.rel_attn.r requires_grad= False
transformer.layer.21.rel_attn.r_r_bias requires_grad= False
transformer.layer.21.rel_attn.r_s_bias requires_grad= False
transformer.layer.21.rel_attn.r_w_bias requires_grad= False
transformer.layer.21.rel_attn.seg_embed requires_grad= False
transformer.layer.21.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.21.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.21.ff.layer_norm.weight requires_grad= False
transformer.layer.21.ff.layer_norm.bias requires_grad= False
transformer.layer.21.ff.layer_1.weight requires_grad= False
transformer.layer.21.ff.layer_1.bias requires_grad= False
transformer.layer.21.ff.layer_2.weight requires_grad= False
transformer.layer.21.ff.layer_2.bias requires_grad= False
transformer.layer.22.rel_attn.q requires_grad= False
transformer.layer.22.rel_attn.k requires_grad= False
transformer.layer.22.rel_attn.v requires_grad= False
transformer.layer.22.rel_attn.o requires_grad= False
transformer.layer.22.rel_attn.r requires_grad= False
transformer.layer.22.rel_attn.r_r_bias requires_grad= False
transformer.layer.22.rel_attn.r_s_bias requires_grad= False
transformer.layer.22.rel_attn.r_w_bias requires_grad= False
transformer.layer.22.rel_attn.seg_embed requires_grad= False
transformer.layer.22.rel_attn.layer_norm.weight requires_grad= False
transformer.layer.22.rel_attn.layer_norm.bias requires_grad= False
transformer.layer.22.ff.layer_norm.weight requires_grad= False
transformer.layer.22.ff.layer_norm.bias requires_grad= False
transformer.layer.22.ff.layer_1.weight requires_grad= False
transformer.layer.22.ff.layer_1.bias requires_grad= False
transformer.layer.22.ff.layer_2.weight requires_grad= False
```

```
transformer.layer.22.ff.layer_2.bias requires_grad= False
transformer.layer.23.rel_attn.q requires_grad= True
transformer.layer.23.rel_attn.k requires_grad= True
transformer.layer.23.rel_attn.v requires_grad= True
transformer.layer.23.rel_attn.o requires_grad= True
transformer.layer.23.rel_attn.r requires_grad= True
transformer.layer.23.rel_attn.r_r_bias requires_grad= True
transformer.layer.23.rel_attn.r_s_bias requires_grad= True
transformer.layer.23.rel_attn.r_w_bias requires_grad= True
transformer.layer.23.rel_attn.seg_embed requires_grad= True
transformer.layer.23.rel_attn.layer_norm.weight requires_grad= True
transformer.layer.23.rel_attn.layer_norm.bias requires_grad= True
transformer.layer.23.ff.layer_norm.weight requires_grad= True
transformer.layer.23.ff.layer_norm.bias requires_grad= True
transformer.layer.23.ff.layer_1.weight requires_grad= True
transformer.layer.23.ff.layer_1.bias requires_grad= True
transformer.layer.23.ff.layer_2.weight requires_grad= True
transformer.layer.23.ff.layer_2.bias requires_grad= True
sequence_summary.summary.weight requires_grad= True
sequence_summary.summary.bias requires_grad= True
logits_proj.weight requires_grad= True
logits_proj.bias requires_grad= True

model.safetensors:   0%|          | 0.00/1.44G [00:00<?, ?B/s]
```

[106]:
```python
# Train & Evaluate
trained_model, trainer_obj = train_transformer_model(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_data_hf,
    val_dataset = val_data_hf,
    output_dir = dir_results,
    num_epochs = num_epochs,
    batch_size = size_batch,
    lr = learning_rate,
    weight_decay = regularization_weight_decay)
metrics = trainer_obj.evaluate()
print("Validation metrics:", metrics)
test_metrics = trainer_obj.evaluate(test_data_hf) if test_data_hf else None
print("Test metrics:", test_metrics)
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611:
FutureWarning: `evaluation_strategy` is deprecated and will be removed in
version 4.46 of  Transformers. Use `eval_strategy` instead
  warnings.warn(
<ipython-input-88-0816cb2c5003>:31: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Validation metrics: {'eval_loss': 0.7015679478645325, 'eval_accuracy':
0.5154394299287411, 'eval_precision': 0.4719626168224299, 'eval_recall':
0.5260416666666666, 'eval_f1': 0.4975369458128079, 'eval_runtime': 5.594,
'eval_samples_per_second': 75.26, 'eval_steps_per_second': 0.715, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.6990423798561096, 'eval_accuracy':
0.5038167938931297, 'eval_precision': 0.48541666666666666, 'eval_recall':
0.528344671201814, 'eval_f1': 0.505971769815418, 'eval_runtime': 7.441,
'eval_samples_per_second': 123.236, 'eval_steps_per_second': 1.075, 'epoch':
1.0}
```

[107]:
```
# save model checkpoint
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
model_save_path = os.path.join(dir_models,␣
 ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
trainer_obj.save_model(model_save_path)
print(f"Model checkpoint saved to: {model_save_path}")
# log experiment results
experiment_info = {
    "model_name": named_model,
    "learning_rate": learning_rate,
    "epochs": num_epochs,
    "batch_size": size_batch,
    "weight_decay": regularization_weight_decay,
    "x_task": x_task,
    "x_col": x_col,
    "y_col": y_col,
    "layers_to_unfreeze": layers_to_unfreeze}
model_info = gather_model_details(trained_model)
all_run_metrics = gather_all_run_metrics(
    trainer=trainer_obj,
    train_dataset=train_data_hf,
    val_dataset=val_data_hf,
    test_dataset=test_data_hf)
log_experiment_results_json(
    experiment_meta=experiment_info,
    model_details=model_info,
    run_metrics=all_run_metrics,
    log_file=log_filepath)
print(f"EXPERIMENT LOGGED TO: {log_filepath}")
```

```
Model checkpoint saved to:
/content/drive/MyDrive/266-final/models/single_xlnet/xlnet-large-
cased_binary_complexity_20250411_122449
```

```
<IPython.core.display.HTML object>
```

```
EXPERIMENT LOGGED TO:
/content/drive/MyDrive/266-final/results/experiment_runs.txt
```