3.8.2.7-multi-bc-snc-morph-complexity-value Lexical Complexity Binary Classification Prediction Transformers Modeling

April 13, 2025

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

```
[1]: !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
```

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

Hit:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64
InRelease
Hit:2 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:7 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:8 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy
InRelease
Hit:9 https://r2u.stat.illinois.edu/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.

```
[3]: #@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
```

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

```
[5]: 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).

```
[6]: 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)
```

[7]:
```
wandbai_api_key = ""
```

[8]:
```
!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
```

[9]:
```
!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
```

[10]: `!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
```

[11]:
```python
#@title Import Data
```

[12]:
```python
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"
```

```python
        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 **

```
[13]: 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
```

[14]:
```python
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
```

[15]:
```python
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
```

[16]:
```python
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
```

[17]:
```python
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"]
    }
```

```
[18]: 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.
```

```python
    """
    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
#     }
```

```python
#     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

```python
[19]: # 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
```

11

```python
# 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
```

```
[20]: 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 *

```
[21]: 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_20250411_123321
multi_bert-base-cased_binary_complexity_20250411_123328
multi_bert-base-cased_binary_complexity_20250411_123334
multi_bert-base-cased_binary_complexity_2025-04-11T08:45:37.508277-07:00
multi_bert-base-cased_binary_complexity_2025-04-11T08:49:23.225636-07:00
multi_bert-base-cased_binary_complexity_2025-04-11T08:52:27.244781-07:00
multi_bert-base-cased_binary_complexity_2025-04-11T08:58:48.717503-07:00
multi_bert-base-cased_binary_complexity_2025-04-11T09:02:19.560179-07:00
multi_bert-base-cased_binary_complexity_2025-04-11T09:07:09.012345-07:00
multi_bert-base-cased_binary_complexity_2025-04-11T09:13:09.359874-07:00
multi_bert-base-cased_binary_complexity_2025-04-11T09:16:49.395761-07:00
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_20250411_123609
multi_bert-large-cased_binary_complexity_20250411_123628
multi_bert-large-cased_binary_complexity_20250411_123925
multi_bert-large-cased_binary_complexity_2025-04-11T08:48:18.745348-07:00
multi_bert-large-cased_binary_complexity_2025-04-11T08:54:05.820676-07:00
multi_bert-large-cased_binary_complexity_2025-04-11T09:02:02.177069-07:00
multi_bert-large-cased_binary_complexity_2025-04-11T09:05:06.965345-07:00
multi_bert-large-cased_binary_complexity_2025-04-11T09:09:16.442088-07:00
multi_bert-large-cased_binary_complexity_2025-04-11T09:19:29.633240-07:00
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_20250411_123340
multi_roberta-base_binary_complexity_20250411_123350
multi_roberta-base_binary_complexity_20250411_123353
multi_roberta-base_binary_complexity_20250411_154623
multi_roberta-base_binary_complexity_20250411_154942
multi_roberta-base_binary_complexity_20250411_155244
multi_roberta-base_binary_complexity_20250411_155935
multi_roberta-base_binary_complexity_20250411_160305
multi_roberta-base_binary_complexity_20250411_160729
multi_roberta-base_binary_complexity_20250411_161357
multi_roberta-base_binary_complexity_20250411_161737
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_20250411_123652
multi_roberta-large_binary_complexity_20250411_123711
```

```
multi_roberta-large_binary_complexity_20250411_124008
multi_roberta-large_binary_complexity_20250411_154930
multi_roberta-large_binary_complexity_20250411_155444
multi_roberta-large_binary_complexity_20250411_160310
multi_roberta-large_binary_complexity_20250411_160618
multi_roberta-large_binary_complexity_20250411_160953
multi_roberta-large_binary_complexity_20250411_162033
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_20250411_120754
single_bert-base-cased_binary_complexity_20250411_120814
single_bert-base-cased_binary_complexity_2025-04-11T08:05:13.902582-07:00
single_bert-base-cased_binary_complexity_2025-04-11T08:06:17.534030-07:00
single_bert-base-cased_binary_complexity_2025-04-11T08:07:27.352056-07:00
```

```
single_bert-base-cased_binary_complexity_2025-04-11T08:26:04.979595-07:00
single_bert-base-cased_binary_complexity_2025-04-11T08:26:08.572701-07:00
single_bert-base-cased_binary_complexity_2025-04-11T08:26:30.856843-07:00
single_bert-base-cased_binary_complexity_2025-04-11T08:26:54.451746-07:00
single_bert-base-cased_binary_complexity_2025-04-11T08:46:21.496737-07:00
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_20250411_121219
single_bert-large-cased_binary_complexity_20250411_121349
single_bert-large-cased_binary_complexity_2025-04-11T08:09:12.255934-07:00
single_bert-large-cased_binary_complexity_2025-04-11T08:09:36.911756-07:00
single_bert-large-cased_binary_complexity_2025-04-11T08:10:24.066329-07:00
single_bert-large-cased_binary_complexity_2025-04-11T08:29:30.583138-07:00
single_bert-large-cased_binary_complexity_2025-04-11T08:29:38.951478-07:00
single_bert-large-cased_binary_complexity_2025-04-11T08:32:25.963533-07:00
single_bert-large-cased_binary_complexity_2025-04-11T08:50:30.982987-07:00
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_20250411_120837
single_roberta-base_binary_complexity_20250411_120924
single_roberta-base_binary_complexity_2025-04-11T08:05:56.911365-07:00
single_roberta-base_binary_complexity_2025-04-11T08:07:02.370250-07:00
single_roberta-base_binary_complexity_2025-04-11T08:08:10.921118-07:00
single_roberta-base_binary_complexity_2025-04-11T08:26:49.931833-07:00
single_roberta-base_binary_complexity_2025-04-11T08:26:52.254114-07:00
single_roberta-base_binary_complexity_2025-04-11T08:28:07.892457-07:00
single_roberta-base_binary_complexity_2025-04-11T08:29:29.440485-07:00
single_roberta-base_binary_complexity_2025-04-11T08:47:34.036076-07:00
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_20250411_121440
single_roberta-large_binary_complexity_20250411_121538
single_roberta-large_binary_complexity_2025-04-11T08:11:00.629981-07:00
single_roberta-large_binary_complexity_2025-04-11T08:11:25.589499-07:00
single_roberta-large_binary_complexity_2025-04-11T08:12:13.265025-07:00
single_roberta-large_binary_complexity_2025-04-11T08:31:21.884000-07:00
single_roberta-large_binary_complexity_2025-04-11T08:31:31.124508-07:00
single_roberta-large_binary_complexity_2025-04-11T08:34:43.895671-07:00
single_roberta-large_binary_complexity_2025-04-11T08:52:46.787583-07:00
single_roberta-large_binary_complexity_75th_split_20250411_010518
single_xlnet
```

[22]:
```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 **

```
[23]:  # 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 **

```
[24]: # 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]), '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])}
```

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

```python
[25]: # 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"
      # x_col = "snc_pos_seq"
      # x_col = "snc_pos_alt"
      # x_col = "snc_morph_seq"
      # x_col = "snc_morph_alt"
      # x_col = "snc_dep_seq"
      # x_col = "snc_dep_alt"
      x_col = "snc_morph_complexity_value"
      ############
      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
```

```python
    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("=============")
```

```python
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%|          | 0/1517 [00:00<?, ? examples/s]

Map:   0%|          | 0/99 [00:00<?, ? examples/s]

Map:   0%|          | 0/184 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([  101,  1573,  1113,  1103,  1397,
 1285,   117,  1165,   138,  1403,
        16669,  4163,  1105, 17666,  4396,  1125,  1435,  1114,  1632,   185,
         4165,  1643,   117,  1105,  1152,  1125,  2242,  1154,  1103,  1282,
         1104,  4510,  1114,  1103,  9463,  3099,  1105,  3981,  1441,  1104,
         1103,  1331,   117,  1120,  1103,  2663,  1104, 22305,  1361,   117,
         1795,  1108,  1814,  1107,   119,   122,   119,  1765, 27677,  1580,
         1571,  1559, 25041,  1545, 18910,  1604, 24050,  1475,   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]),
'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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])}
```

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 16:16:28'}
=============
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: multi
input column: snc_morph_complexity_value
```

[26]:
```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>

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

<IPython.core.display.HTML object>

Validation metrics: {'eval_loss': 0.6908776164054871, 'eval_accuracy':
0.5151515151515151, 'eval_precision': 0.5157894736842106, 'eval_recall':
0.9607843137254902, 'eval_f1': 0.6712328767123288, 'eval_runtime': 4.9292,
'eval_samples_per_second': 20.084, 'eval_steps_per_second': 0.203, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.6944844722747803, 'eval_accuracy':
0.5108695652173914, 'eval_precision': 0.5254237288135594, 'eval_recall':
0.9393939393939394, 'eval_f1': 0.6739130434782609, 'eval_runtime': 5.218,
'eval_samples_per_second': 35.263, 'eval_steps_per_second': 0.383, 'epoch': 1.0}

```python
[27]: # save model checkpoint
      # timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
      pacific_time = datetime.now(zoneinfo.ZoneInfo("America/Los_Angeles"))
      timestamp = pacific_time.isoformat()
      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/multi_bert-

base-cased_binary_complexity_2025-04-11T09:16:49.395761-07:00

<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

```python
[28]:  # 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"
       # x_col = "snc_pos_seq"
       # x_col = "snc_pos_alt"
       # x_col = "snc_morph_seq"
       # x_col = "snc_morph_alt"
       # x_col = "snc_dep_seq"
       # x_col = "snc_dep_alt"
       x_col = "snc_morph_complexity_value"
       ############
       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
```

```python
    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%| | 0/1517 [00:00<?, ? examples/s]

Map: 0%| | 0/99 [00:00<?, ? examples/s]

Map: 0%| | 0/184 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([101, 1573, 1113, 1103, 1397,
1285,   117, 1165,   138, 1403,
        16669, 4163, 1105, 17666, 4396, 1125, 1435, 1114, 1632,   185,
         4165, 1643,   117, 1105, 1152, 1125, 2242, 1154, 1103, 1282,
         1104, 4510, 1114, 1103, 9463, 3099, 1105, 3981, 1441, 1104,
         1103, 1331,   117, 1120, 1103, 2663, 1104, 22305, 1361,   117,
         1795, 1108, 1814, 1107,   119,   122,   119, 1765, 27677, 1580,
         1571, 1559, 25041, 1545, 18910, 1604, 24050, 1475,   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]),
'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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])}
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 16:17:18'}
============
```

[29]: `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)
  )
)
```

```
[30]: 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
```

[31]:
```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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 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])
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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 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, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
       0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 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
[32]: 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: multi
input column: snc_morph_complexity_value
```

```python
[33]: 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
```

[34]:
```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': 0.6930475831031799, 'eval_accuracy':
0.5151515151515151, 'eval_precision': 0.5151515151515151, 'eval_recall': 1.0,
'eval_f1': 0.68, 'eval_runtime': 4.934, 'eval_samples_per_second': 20.065,
'eval_steps_per_second': 0.203, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.6915158629417419, 'eval_accuracy':
0.5380434782608695, 'eval_precision': 0.5380434782608695, 'eval_recall': 1.0,
'eval_f1': 0.6996466431095406, 'eval_runtime': 5.0241,
'eval_samples_per_second': 36.623, 'eval_steps_per_second': 0.398, 'epoch': 1.0}

```
[35]: # 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/multi_roberta-
base_binary_complexity_20250411_161737

<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

```
[25]: # 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
```

```python
############
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"
# x_col = "snc_pos_seq"
# x_col = "snc_pos_alt"
# x_col = "snc_morph_seq"
# x_col = "snc_morph_alt"
# x_col = "snc_dep_seq"
# x_col = "snc_dep_alt"
x_col = "snc_morph_complexity_value"
############
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="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%|              | 0/1517 [00:00<?, ? examples/s]

Map:   0%|              | 0/99 [00:00<?, ? examples/s]

Map:   0%|              | 0/184 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([  101,  1573,  1113,  1103,  1397,
1285,   117,  1165,   138,  1403,
       16669,  4163,  1105, 17666,  4396,  1125,  1435,  1114,  1632,   185,
        4165,  1643,   117,  1105,  1152,  1125,  2242,  1154,  1103,  1282,
        1104,  4510,  1114,  1103,  9463,  3099,  1105,  3981,  1441,  1104,
        1103,  1331,   117,  1120,  1103,  2663,  1104, 22305,  1361,   117,
        1795,  1108,  1814,  1107,   119,   122,   119,  1765, 27677,  1580,
```

```
       1571,  1559, 25041,  1545, 18910,  1604, 24050,  1475,   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]),
 '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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])}
Loading from Hugging Face model: bert-large-cased

tokenizer_config.json:   0%|          | 0.00/49.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/762 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/213k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/436k [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/1.34G [00:00<?, ?B/s]

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 16:19:03'}
=============
```

[26]:
```python
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)
)
```

```python
[27]: 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: multi
input column: snc_morph_complexity_value
=============
num_trainable_parameters: 13647874
```

[29]:
```python
model.resize_token_embeddings(len(tokenizer))
```

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

[30]:
```python
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
```

[31]:
```python
model.resize_token_embeddings(len(tokenizer))
```

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

[32]:
```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>

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

<IPython.core.display.HTML object>

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Validation metrics: {'eval_loss': 0.7986159920692444, 'eval_accuracy':
0.48484848484848486, 'eval_precision': 0.0, 'eval_recall': 0.0, 'eval_f1': 0.0,
'eval_runtime': 5.6558, 'eval_samples_per_second': 17.504,
'eval_steps_per_second': 0.177, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.8214293718338013, 'eval_accuracy':
0.46195652173913043, 'eval_precision': 0.0, 'eval_recall': 0.0, 'eval_f1': 0.0,

```

```
'eval_runtime': 6.1565, 'eval_samples_per_second': 29.887,
'eval_steps_per_second': 0.325, 'epoch': 1.0}
```

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

[33]:
```python
# save model checkpoint
# timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
pacific_time = datetime.now(zoneinfo.ZoneInfo("America/Los_Angeles"))
timestamp = pacific_time.isoformat()
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/multi_bert-
large-cased_binary_complexity_2025-04-11T09:19:29.633240-07:00

<IPython.core.display.HTML object>

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

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

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

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

```python
[34]: # 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"
      # x_col = "snc_pos_seq"
      # x_col = "snc_pos_alt"
      # x_col = "snc_morph_seq"
      # x_col = "snc_morph_alt"
      # x_col = "snc_dep_seq"
      # x_col = "snc_dep_alt"
      x_col = "snc_morph_complexity_value"
      ############
      y_col = "binary_complexity"
      # y_col = "complexity"
      ############
      # x_task = "single"
      x_task = "multi"
      if x_task == "single":
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

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

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

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

```python
[34]: # 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"
      # x_col = "snc_pos_seq"
      # x_col = "snc_pos_alt"
      # x_col = "snc_morph_seq"
      # x_col = "snc_morph_alt"
      # x_col = "snc_dep_seq"
      # x_col = "snc_dep_alt"
      x_col = "snc_morph_complexity_value"
      ############
      y_col = "binary_complexity"
      # y_col = "complexity"
      ############
      # x_task = "single"
      x_task = "multi"
      if x_task == "single":
```

```python
    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%| | 0/1517 [00:00<?, ? examples/s]

Map: 0%| | 0/99 [00:00<?, ? examples/s]

Map: 0%| | 0/184 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([101,   1573,   1113,   1103,   1397,
1285,   117,   1165,   138,   1403,
        16669,   4163,   1105, 17666,   4396,   1125,   1435,   1114,   1632,   185,
         4165,   1643,   117,   1105,   1152,   1125,   2242,   1154,   1103,   1282,
         1104,   4510,   1114,   1103,   9463,   3099,   1105,   3981,   1441,   1104,
         1103,   1331,   117,   1120,   1103,   2663,   1104, 22305,   1361,   117,
         1795,   1108,   1814,   1107,   119,   122,   119,   1765, 27677,   1580,
         1571,   1559, 25041,   1545, 18910,   1604, 24050,   1475,   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]),
'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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])}
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 16:20:08'}
=============
```

[35]: `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)
          )
        )
```

```python
[36]: 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
```

[37]:
```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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 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])
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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 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, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 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
[38]:  #########################################
       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:")
       print("model used:", named_model)
       print("learning rate used:", learning_rate)
```

```python
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: multi
input column: snc_morph_complexity_value
=============
num_trainable_parameters: 13647874
```

[39]:
```python
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
```

[40]:
```python
model.resize_token_embeddings(len(tokenizer))
```

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

[41]:
```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': 0.6942838430404663, 'eval_accuracy':
0.5151515151515151, 'eval_precision': 0.5151515151515151, 'eval_recall': 1.0,
'eval_f1': 0.68, 'eval_runtime': 5.7156, 'eval_samples_per_second': 17.321,
'eval_steps_per_second': 0.175, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.6916723847389221, 'eval_accuracy':
0.5380434782608695, 'eval_precision': 0.5380434782608695, 'eval_recall': 1.0,
'eval_f1': 0.6996466431095406, 'eval_runtime': 6.1884,
'eval_samples_per_second': 29.733, 'eval_steps_per_second': 0.323, 'epoch': 1.0}

```python
[42]: # 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/multi_roberta-
large_binary_complexity_20250411_162033

<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
[43]: # 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"
      # x_col = "snc_pos_seq"
      # x_col = "snc_pos_alt"
      # x_col = "snc_morph_seq"
      # x_col = "snc_morph_alt"
      # x_col = "snc_dep_seq"
      # x_col = "snc_dep_alt"
      x_col = "snc_morph_complexity_value"
      ###########
      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
```

```python
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%|          | 0/1517 [00:00<?, ? examples/s]
```

```
Map:    0%|              | 0/99 [00:00<?, ? examples/s]

Map:    0%|              | 0/184 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([    0,  2847,    15,     5,   220,
183,     6,    77,  3303, 22872,
          102,     8,  6552,  2463,    56,   283,    19,   372, 34415,     6,
            8,    51,    56,  2867,    88,     5,   317,     9,  1576,    19,
            5, 20510,  1024,     8,  5402,   604,     9,     5,   343,     6,
           23,     5,  5936,     9, 13326,   687,     6,  1206,    21,  1146,
           11,     4,   112,     4,  2518, 38223,  4390, 33973,  2940,  4531,
         1225,     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]),
'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, 1, 1, 1, 1, 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])}
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 16:21:08'}
=============
```

[44]: `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)
  )
```

```
[45]: 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
```

[46]:
```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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])
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])
tensor([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])
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])
tensor([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])
```

[47]:
```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",
```

```python
        "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: multi
input column: snc_morph_complexity_value
=============
num_trainable_parameters: 5607938
```

```python
[48]: 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
```

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

```python
[50]:  # 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': 2.441112756729126, 'eval_accuracy':
0.5151515151515151, 'eval_precision': 0.5151515151515151, 'eval_recall': 1.0,
'eval_f1': 0.68, 'eval_runtime': 5.6281, 'eval_samples_per_second': 17.59,
'eval_steps_per_second': 0.178, 'epoch': 1.0}
Test metrics: {'eval_loss': 2.0650463104248047, 'eval_accuracy':
0.5380434782608695, 'eval_precision': 0.5380434782608695, 'eval_recall': 1.0,
'eval_f1': 0.6996466431095406, 'eval_runtime': 5.1594,
'eval_samples_per_second': 35.663, 'eval_steps_per_second': 0.388, 'epoch': 1.0}

```
[51]: # save model checkpoint
      # timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
      pacific_time = datetime.now(zoneinfo.ZoneInfo("America/Los_Angeles"))
      timestamp = pacific_time.isoformat()
      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}")
```

```python
# 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/multi_answerdotai/ModernBERT-
base_binary_complexity_2025-04-11T09:21:47.985625-07:00

<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

```python
[52]: # 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
```

```python
#########################################
# x_col = "sentence"
# x_col = "sentence_no_contractions"
# x_col = "pos_sequence"
# x_col = "dep_sequence"
# x_col = "morph_sequence"
# x_col = "snc_pos_seq"
# x_col = "snc_pos_alt"
# x_col = "snc_morph_seq"
# x_col = "snc_morph_alt"
# x_col = "snc_dep_seq"
# x_col = "snc_dep_alt"
x_col = "snc_morph_complexity_value"
############
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)
```

```python
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%|          | 0/1517 [00:00<?, ? examples/s]

Map:    0%|          | 0/99 [00:00<?, ? examples/s]

Map:    0%|          | 0/184 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([50281,  2598,   327,   253,  1735,
1388,    13,   672,  3419, 21743,
           66,   285, 10246,   547,   574,  1705,   342,  1270,   268, 23242,
           13,   285,   597,   574,  5966,   715,   253,  1659,   273,  4854,
          342,   253, 36174,  6251,   285,  8624,  1821,   273,   253,  2846,
           13,   387,   253,  3923,   273, 43149,   316,    13,  5171,   369,
         3982,   275,    15,   337,    15,  1630,  2082, 34920, 27590,  1438,
         2227,   883, 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]),
'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, 1, 1, 1, 1, 1, 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])}
```

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 16:22:18'}
============

```
[53]: 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)
)
```

```
[54]: 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
```

[55]:
```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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])
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, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 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, 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])
```

```python
[56]: #########################################
      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: multi
input column: snc_morph_complexity_value
=============
num_trainable_parameters: 13309954
```

```python
[57]: 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
```

[58]:
```python
# model.resize_token_embeddings(len(tokenizer))
```

```
[59]:   # 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 16:22:33.480000 3862 torch/_dynamo/convert_frame.py:906] [1/8]
torch._dynamo hit config.cache_size_limit (8)
W0411 16:22:33.480000 3862 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 16:22:33.480000 3862 torch/_dynamo/convert_frame.py:906] [1/8]   last
reason: 1/0: GLOBAL_STATE changed: grad_mode
W0411 16:22:33.480000 3862 torch/_dynamo/convert_frame.py:906] [1/8] To log all
recompilation reasons, use TORCH_LOGS="recompiles".
W0411 16:22:33.480000 3862 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': 0.7499406337738037, 'eval_accuracy':
0.5454545454545454, 'eval_precision': 0.5319148936170213, 'eval_recall':
0.9803921568627451, 'eval_f1': 0.6896551724137931, 'eval_runtime': 5.7123,
'eval_samples_per_second': 17.331, 'eval_steps_per_second': 0.175, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.7823100090026855, 'eval_accuracy':
0.5217391304347826, 'eval_precision': 0.5321637426900585, 'eval_recall':
0.9191919191919192, 'eval_f1': 0.674074074074074, 'eval_runtime': 5.7542,
'eval_samples_per_second': 31.977, 'eval_steps_per_second': 0.348, 'epoch': 1.0}

128

```
[60]:  # 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/multi_answerdotai/ModernBERT-
large_binary_complexity_20250411_162250

<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

```
[61]:  # 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
```

```python
# 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"
# x_col = "snc_pos_seq"
# x_col = "snc_pos_alt"
# x_col = "snc_morph_seq"
# x_col = "snc_morph_alt"
# x_col = "snc_dep_seq"
# x_col = "snc_dep_alt"
x_col = "snc_morph_complexity_value"
############
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,
```

```python
        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%|            | 0/1517 [00:00<?, ? examples/s]

Map:   0%|            | 0/99 [00:00<?, ? examples/s]

Map:   0%|            | 0/184 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([50281,  2598,   327,   253,  1735,
1388,    13,   672,  3419, 21743,
          66,   285, 10246,   547,   574,  1705,   342,  1270,   268, 23242,
          13,   285,   597,   574,  5966,   715,   253,  1659,   273,  4854,
         342,   253, 36174,  6251,   285,  8624,  1821,   273,   253,  2846,
          13,   387,   253,  3923,   273, 43149,   316,    13,  5171,   369,
        3982,   275,    15,   337,    15,  1630,  2082, 34920, 27590,  1438,
        2227,   883, 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]),
 '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, 1, 1, 1, 1, 1, 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])}
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 16:23:25'}
=============
```

```
[62]: 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)
)
```

```python
[63]: 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
```

[64]:
```python
##########################################
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)
```

```python
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: multi
input column: snc_morph_complexity_value
=============
num_trainable_parameters: 8074754
```

[65]:
```python
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
```

[66]:
```python
# model.resize_token_embeddings(len(tokenizer))
```

[67]:
```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(

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

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Validation metrics: {'eval_loss': 0.6897398829460144, 'eval_accuracy':
0.5454545454545454, 'eval_precision': 0.7142857142857143, 'eval_recall':
0.19607843137254902, 'eval_f1': 0.3076923076923077, 'eval_runtime': 4.914,
'eval_samples_per_second': 20.147, 'eval_steps_per_second': 0.204, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.7092766165733337, 'eval_accuracy':
0.41847826086956524, 'eval_precision': 0.36666666666666664, 'eval_recall':
0.1111111111111111, 'eval_f1': 0.17054263565891473, 'eval_runtime': 5.106,
'eval_samples_per_second': 36.036, 'eval_steps_per_second': 0.392, 'epoch': 1.0}
```

```python
[68]:  # save model checkpoint
       # timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
       pacific_time = datetime.now(zoneinfo.ZoneInfo("America/Los_Angeles"))
       timestamp = pacific_time.isoformat()
       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/multi_microsoft/deberta-v3-base_binary_complexity_2025-04-
11T09:23:48.387428-07:00

<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

```python
[25]:  # 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"
       # x_col = "snc_pos_seq"
       # x_col = "snc_pos_alt"
       # x_col = "snc_morph_seq"
       # x_col = "snc_morph_alt"
       # x_col = "snc_dep_seq"
       # x_col = "snc_dep_alt"
       x_col = "snc_morph_complexity_value"
       ###########
       y_col = "binary_complexity"
       # y_col = "complexity"
```

146

```python
############
# 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%| | 0/1517 [00:00<?, ? examples/s]

Map: 0%| | 0/99 [00:00<?, ? examples/s]

Map: 0%| | 0/184 [00:00<?, ? examples/s]

Datasets prepared. Sample from train_data_hf:
 {'labels': tensor(0), 'input_ids': tensor([101, 1573, 1113, 1103, 1397,
 1285,   117, 1165,   138, 1403,
        16669, 4163, 1105, 17666, 4396, 1125, 1435, 1114, 1632,   185,
         4165, 1643,   117, 1105, 1152, 1125, 2242, 1154, 1103, 1282,
         1104, 4510, 1114, 1103, 9463, 3099, 1105, 3981, 1441, 1104,
         1103, 1331,   117, 1120, 1103, 2663, 1104, 22305, 1361,   117,
         1795, 1108, 1814, 1107,   119,   122,   119, 1765, 27677, 1580,
         1571, 1559, 25041, 1545, 18910, 1604, 24050, 1475,   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]),
'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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])}
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 16:25:15'}
=============
```

[26]: `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)
)
```

[27]:
```
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
```

[28]:
```python
########################################
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:")
```

```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))
```

```
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: multi
input column: snc_morph_complexity_value
============
num_trainable_parameters: 8270594
```

```python
[29]: 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
```

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

```python
[31]: # 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>

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

<IPython.core.display.HTML object>

Validation metrics: {'eval_loss': 0.7216587066650391, 'eval_accuracy':
0.5151515151515151, 'eval_precision': 0.5151515151515151, 'eval_recall': 1.0,
'eval_f1': 0.68, 'eval_runtime': 5.4811, 'eval_samples_per_second': 18.062,
'eval_steps_per_second': 0.182, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.7248739004135132, 'eval_accuracy':
0.5380434782608695, 'eval_precision': 0.5380434782608695, 'eval_recall': 1.0,
'eval_f1': 0.6996466431095406, 'eval_runtime': 5.2777,
'eval_samples_per_second': 34.864, 'eval_steps_per_second': 0.379, 'epoch': 1.0}

```python
[32]: # save model checkpoint
      # timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
      pacific_time = datetime.now(zoneinfo.ZoneInfo("America/Los_Angeles"))
      timestamp = pacific_time.isoformat()
      model_save_path = os.path.join(dir_models,␣
        ↪f"{x_task}_{named_model}_{y_col}_{timestamp}")
      trainer_obj.save_model(model_save_path)
```

```python
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/multi_xlnet/xlnet-base-
cased_binary_complexity_2025-04-11T09:25:38.158497-07:00

<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

```python
[33]: # 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" #
###########
```

```python
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"
# x_col = "snc_pos_seq"
# x_col = "snc_pos_alt"
# x_col = "snc_morph_seq"
# x_col = "snc_morph_alt"
# x_col = "snc_dep_seq"
# x_col = "snc_dep_alt"
x_col = "snc_morph_complexity_value"
############
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%|              | 0/1517 [00:00<?, ? examples/s]

Map:    0%|              | 0/99 [00:00<?, ? examples/s]

Map:    0%|              | 0/184 [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,    346,     31,     18,    244,    191,
         19,     90, 26172, 22226,     21, 10839,   2960,     54,    280,     33,
        312, 30029,     19,     21,     63,     54,   2000,     91,     18,    250,
         20,   2243,     33,     18, 15010,   1794,     21,   3824,    416,     20,
         18,    285,     19,     38,     18,   2217,     20,     17, 20610,    415,
         19,   1209,     30,    959,     25,      9,    156,      9,   2765,   3802,
```

```
              2270,  5185,  3903,  2728,  4235,  1545,     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, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
              1, 1, 1, 1, 1, 1, 1, 1, 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 16:26:14'}
=============

[34]: 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)
)
```

[35]:
```python
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

[36]:   ########################################
        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: multi
input column: snc_morph_complexity_value
=============
num_trainable_parameters: 14697474
```

```python
[37]: 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
```

```python
[38]: # 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%| | 0.00/1.44G [00:00<?, ?B/s]

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Validation metrics: {'eval_loss': 0.6860107779502869, 'eval_accuracy':
0.5555555555555556, 'eval_precision': 0.5813953488372093, 'eval_recall':
0.49019607843137253, 'eval_f1': 0.5319148936170213, 'eval_runtime': 5.4574,
'eval_samples_per_second': 18.141, 'eval_steps_per_second': 0.183, 'epoch': 1.0}
Test metrics: {'eval_loss': 0.6885326504707336, 'eval_accuracy':
0.5815217391304348, 'eval_precision': 0.6170212765957447, 'eval_recall':
0.5858585858585859, 'eval_f1': 0.6010362694300518, 'eval_runtime': 5.9769,
'eval_samples_per_second': 30.785, 'eval_steps_per_second': 0.335, 'epoch': 1.0}

```python
[39]: # save model checkpoint
      # timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
      pacific_time = datetime.now(zoneinfo.ZoneInfo("America/Los_Angeles"))
      timestamp = pacific_time.isoformat()
      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
```

```python
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/multi_xlnet/xlnet-large-cased_binary_complexity_2025-04-11T09:26:43.615411-07:00

<IPython.core.display.HTML object>

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