```
!pip install evaluate
!pip install -U transformers
!pip install bert_score
!pip install rouge_score
```

```
Requirement already satisfied: evaluate in
/usr/local/lib/python3.12/dist-packages (0.4.6)
Requirement already satisfied: datasets>=2.0.0 in
/usr/local/lib/python3.12/dist-packages (from evaluate) (4.3.0)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.12/dist-packages (from evaluate) (2.0.2)
Requirement already satisfied: dill in /usr/local/lib/python3.12/dist-
packages (from evaluate) (0.3.8)
Requirement already satisfied: pandas in
/usr/local/lib/python3.12/dist-packages (from evaluate) (2.2.2)
Requirement already satisfied: requests>=2.19.0 in
/usr/local/lib/python3.12/dist-packages (from evaluate) (2.32.4)
Requirement already satisfied: tqdm>=4.62.1 in
/usr/local/lib/python3.12/dist-packages (from evaluate) (4.67.1)
Requirement already satisfied: xxhash in
/usr/local/lib/python3.12/dist-packages (from evaluate) (3.6.0)
Requirement already satisfied: multiprocess in
/usr/local/lib/python3.12/dist-packages (from evaluate) (0.70.16)
Requirement already satisfied: fsspec>=2021.05.0 in
/usr/local/lib/python3.12/dist-packages (from fsspec[http]>=2021.05.0-
>evaluate) (2025.3.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in
/usr/local/lib/python3.12/dist-packages (from evaluate) (0.36.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.12/dist-packages (from evaluate) (25.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from datasets>=2.0.0-
>evaluate) (3.20.0)
Requirement already satisfied: pyarrow>=21.0.0 in
/usr/local/lib/python3.12/dist-packages (from datasets>=2.0.0-
>evaluate) (22.0.0)
Requirement already satisfied: httpx<1.0.0 in
/usr/local/lib/python3.12/dist-packages (from datasets>=2.0.0-
>evaluate) (0.28.1)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.12/dist-packages (from datasets>=2.0.0-
>evaluate) (6.0.3)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in
/usr/local/lib/python3.12/dist-packages (from fsspec[http]>=2021.05.0-
>evaluate) (3.13.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.7.0-
>evaluate) (4.15.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.7.0-
```

```
>evaluate) (1.2.0)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests>=2.19.0-
>evaluate) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests>=2.19.0-
>evaluate) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests>=2.19.0-
>evaluate) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests>=2.19.0-
>evaluate) (2025.10.5)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas->evaluate)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.12/dist-packages (from pandas->evaluate)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.12/dist-packages (from pandas->evaluate)
(2025.2)
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (2.6.1)
Requirement already satisfied: aiosignal>=1.4.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (1.4.0)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (25.4.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (1.8.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (6.7.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (0.4.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!
=4.0.0a1->fsspec[http]>=2021.05.0->evaluate) (1.22.0)
Requirement already satisfied: anyio in
/usr/local/lib/python3.12/dist-packages (from httpx<1.0.0-
>datasets>=2.0.0->evaluate) (4.11.0)
Requirement already satisfied: httpcore==1.* in
/usr/local/lib/python3.12/dist-packages (from httpx<1.0.0-
>datasets>=2.0.0->evaluate) (1.0.9)
```

```
Requirement already satisfied: h11>=0.16 in
/usr/local/lib/python3.12/dist-packages (from httpcore==1.*-
>httpx<1.0.0->datasets>=2.0.0->evaluate) (0.16.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2-
>pandas->evaluate) (1.17.0)
Requirement already satisfied: sniffio>=1.1 in
/usr/local/lib/python3.12/dist-packages (from anyio->httpx<1.0.0-
>datasets>=2.0.0->evaluate) (1.3.1)
Requirement already satisfied: transformers in
/usr/local/lib/python3.12/dist-packages (4.57.1)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from transformers) (3.20.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in
/usr/local/lib/python3.12/dist-packages (from transformers) (0.36.0)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.12/dist-packages (from transformers) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.12/dist-packages (from transformers)
(2024.11.6)
Requirement already satisfied: requests in
/usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in
/usr/local/lib/python3.12/dist-packages (from transformers) (0.22.1)
Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in
/usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.12/dist-packages (from huggingface-
hub<1.0,>=0.34.0->transformers) (2025.3.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-
hub<1.0,>=0.34.0->transformers) (4.15.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-
hub<1.0,>=0.34.0->transformers) (1.2.0)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers)
(3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers)
(3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers)
```

```
(2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers)
(2025.10.5)
Requirement already satisfied: bert_score in
/usr/local/lib/python3.12/dist-packages (0.3.13)
Requirement already satisfied: torch>=1.0.0 in
/usr/local/lib/python3.12/dist-packages (from bert_score)
(2.8.0+cu126)
Requirement already satisfied: pandas>=1.0.1 in
/usr/local/lib/python3.12/dist-packages (from bert_score) (2.2.2)
Requirement already satisfied: transformers>=3.0.0 in
/usr/local/lib/python3.12/dist-packages (from bert_score) (4.57.1)
Requirement already satisfied: numpy in
/usr/local/lib/python3.12/dist-packages (from bert_score) (2.0.2)
Requirement already satisfied: requests in
/usr/local/lib/python3.12/dist-packages (from bert_score) (2.32.4)
Requirement already satisfied: tqdm>=4.31.1 in
/usr/local/lib/python3.12/dist-packages (from bert_score) (4.67.1)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.12/dist-packages (from bert_score) (3.10.0)
Requirement already satisfied: packaging>=20.9 in
/usr/local/lib/python3.12/dist-packages (from bert_score) (25.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas>=1.0.1-
>bert_score) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.12/dist-packages (from pandas>=1.0.1-
>bert_score) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.12/dist-packages (from pandas>=1.0.1-
>bert_score) (2025.2)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (3.20.0)
Requirement already satisfied: typing-extensions>=4.10.0 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (4.15.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (1.13.3)
Requirement already satisfied: networkx in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (3.5)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
```

```
>bert_score) (3.1.6)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.0.0-
>bert_score) (3.4.0)
```

```
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in
/usr/local/lib/python3.12/dist-packages (from transformers>=3.0.0-
>bert_score) (0.36.0)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.12/dist-packages (from transformers>=3.0.0-
>bert_score) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.12/dist-packages (from transformers>=3.0.0-
>bert_score) (2024.11.6)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in
/usr/local/lib/python3.12/dist-packages (from transformers>=3.0.0-
>bert_score) (0.22.1)
Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.12/dist-packages (from transformers>=3.0.0-
>bert_score) (0.6.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->bert_score)
(1.3.3)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->bert_score)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->bert_score)
(4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->bert_score)
(1.4.9)
Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->bert_score)
(11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->bert_score)
(3.2.5)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests->bert_score)
(3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests->bert_score)
(3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests->bert_score)
(2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests->bert_score)
(2025.10.5)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-
hub<1.0,>=0.34.0->transformers>=3.0.0->bert_score) (1.2.0)
Requirement already satisfied: six>=1.5 in
```

```
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2-
>pandas>=1.0.1->bert_score) (1.17.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3-
>torch>=1.0.0->bert_score) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->torch>=1.0.0-
>bert_score) (3.0.3)
Requirement already satisfied: rouge_score in
/usr/local/lib/python3.12/dist-packages (0.1.2)
Requirement already satisfied: absl-py in
/usr/local/lib/python3.12/dist-packages (from rouge_score) (1.4.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-
packages (from rouge_score) (3.9.1)
Requirement already satisfied: numpy in
/usr/local/lib/python3.12/dist-packages (from rouge_score) (2.0.2)
Requirement already satisfied: six>=1.14.0 in
/usr/local/lib/python3.12/dist-packages (from rouge_score) (1.17.0)
Requirement already satisfied: click in
/usr/local/lib/python3.12/dist-packages (from nltk->rouge_score)
(8.3.0)
Requirement already satisfied: joblib in
/usr/local/lib/python3.12/dist-packages (from nltk->rouge_score)
(1.5.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.12/dist-packages (from nltk->rouge_score)
(2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-
packages (from nltk->rouge_score) (4.67.1)
```

```python
import os
import json
import random
from dataclasses import dataclass
from typing import Dict, List, Optional

import numpy as np
import torch

import transformers

from datasets import Dataset, DatasetDict
from transformers import (
    T5TokenizerFast,
    T5ForConditionalGeneration,
    TrainingArguments,
    Trainer,
    DataCollatorForSeq2Seq,
    set_seed,
)
```

```
import evaluate

#!/usr/bin/env python3
SEED = 42
set_seed(SEED)

MODEL_NAME = "t5-base"
MAX_SOURCE_LENGTH = 256
MAX_TARGET_LENGTH = 64
TRAIN_BATCH_SIZE = 8
EVAL_BATCH_SIZE = 8
NUM_EPOCHS = 8
LR = 5e-5
WEIGHT_DECAY = 0.01
OUTPUT_DIR =
"/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints"
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"

TRAIN_FILE = "/content/drive/MyDrive/DeepLearning/train.json"
VALID_FILE = "/content/drive/MyDrive/DeepLearning/validation.json"
TEST_FILE  = "/content/drive/MyDrive/DeepLearning/test.json"

# ------------------------------
# Helpers to load the JSON files
# ------------------------------
def load_json_records(path: str) -> List[Dict]:
    """
    Load local json file that may be either:
      - a dict mapping id -> record
      - a list of record dicts
    Return a list of record dicts.
    """
    with open(path, "r", encoding="utf-8") as f:
        data = json.load(f)
    if isinstance(data, dict):
        # convert to list
        recs = []
        for k, v in data.items():
            # ensure id inside
            if isinstance(v, dict):
                v["_id"] = k
                recs.append(v)
            else:
                # if values are not dicts, try to wrap
                recs.append({"_id": k, "value": v})
        return recs
    elif isinstance(data, list):
        return data
    else:
```

```
        raise ValueError(f"Unsupported JSON structure in {path}")

def extract_question_summary_pairs(recs: List[Dict],
question_field="question", summary_field="multi_abs_summ"):
    out = []
    for r in recs:
        q = r.get(question_field, None)
        s = r.get(summary_field, None)
        if q is None:
            # maybe nested under 'question' key
            continue
        if s is None:
            # If no summary present, skip
            continue
        out.append({"question": q.strip(), "summary": s.strip(),
"_id": r.get("_id", None)})
    return out

# ----------------------------
# Build datasets
# ----------------------------
print("Loading data files...")
train_recs = load_json_records(TRAIN_FILE)
val_recs = load_json_records(VALID_FILE)
test_recs = load_json_records(TEST_FILE)

print(f"Raw records counts: train={len(train_recs)},
valid={len(val_recs)}, test={len(test_recs)}")

train_pairs = extract_question_summary_pairs(train_recs)
val_pairs   = extract_question_summary_pairs(val_recs)
test_pairs  = extract_question_summary_pairs(test_recs)

print(f"Pairs counts (question+summary): train={len(train_pairs)},
valid={len(val_pairs)}, test={len(test_pairs)}")

# Optionally shuffle training
random.shuffle(train_pairs)

ds_train = Dataset.from_list(train_pairs)
ds_val   = Dataset.from_list(val_pairs)
ds_test  = Dataset.from_list(test_pairs)

dataset_dict = DatasetDict({"train": ds_train, "validation": ds_val,
"test": ds_test})

Loading data files...
Raw records counts: train=110, valid=16, test=30
Pairs counts (question+summary): train=110, valid=16, test=30
```

```
# ----------------------------
# Tokenizer & Model
# ----------------------------
print("Loading tokenizer and model:", MODEL_NAME)
tokenizer = T5TokenizerFast.from_pretrained(MODEL_NAME)
model = T5ForConditionalGeneration.from_pretrained(MODEL_NAME)
model.to(DEVICE)

# Prefix for T5 (helps the model know the task)
TASK_PREFIX = "summarize: "

# ----------------------------
# Preprocessing function
# ----------------------------
def preprocess_function(examples):
    inputs = [TASK_PREFIX + ex for ex in examples["question"]]
    model_inputs = tokenizer(
        inputs,
        max_length=MAX_SOURCE_LENGTH,
        truncation=True,
        padding="max_length",
    )

    # Setup the tokenizer for targets
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
            examples["summary"],
            max_length=MAX_TARGET_LENGTH,
            truncation=True,
            padding="max_length",
        )

    # replace tokenizer.pad_token_id in labels by -100 to ignore when
computing loss
    label_ids = labels["input_ids"]
    # convert padding token id's to -100
    label_ids = [
        [(token if token != tokenizer.pad_token_id else -100) for
token in label] for label in label_ids
    ]

    model_inputs["labels"] = label_ids
    return model_inputs

print("Tokenizing datasets...")
tokenized_datasets = dataset_dict.map(
    preprocess_function,
    batched=True,
    remove_columns=dataset_dict["train"].column_names,
)
```

```
# ----------------------------
# Data collator
# ----------------------------
data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)

# ----------------------------
# Metrics setup
# ----------------------------
rouge = evaluate.load("rouge")
bleu = evaluate.load("bleu")
bertscore = evaluate.load("bertscore")

def postprocess_text(preds, labels):
    # decode token ids to strings if needed (but here preds/labels are
strings)
    preds = [p.strip() for p in preds]
    labels = [l.strip() for l in labels]
    # rouge requires newline separated sentences; leave as is
    return preds, labels
```

```
Loading tokenizer and model: t5-base
Tokenizing datasets...
```


```
{"model_id":"b408275a6d054f5593eda0f6ab760970","version_major":2,"vers
ion_minor":0}

{"model_id":"873a51c0d2be4d778856e1c70a10294d","version_major":2,"vers
ion_minor":0}

{"model_id":"d36ab06c98e440c98b5aaea7f43aa296","version_major":2,"vers
ion_minor":0}

{"model_id":"a9ec5ad6b9cc41d0a5bd9aaed811126e","version_major":2,"vers
ion_minor":0}

{"model_id":"300aad811ad143d6bebbd2833b9a8014","version_major":2,"vers
ion_minor":0}

{"model_id":"ddb631e5cac44e2482669ce101856c31","version_major":2,"vers
ion_minor":0}

{"model_id":"2dc43a39e62b40189df8c1cb34057105","version_major":2,"vers
ion_minor":0}
```


```
def compute_metrics(eval_pred):
    preds_ids, labels_ids = eval_pred
    # decode
    if isinstance(preds_ids, tuple):
        preds_ids = preds_ids[0]
    decoded_preds = tokenizer.batch_decode(preds_ids,
```

```
skip_special_tokens=True)
    # labels: replace -100 tokens then decode
    labels_ids = np.where(labels_ids != -100, labels_ids,
tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels_ids,
skip_special_tokens=True)

    # postprocess
    decoded_preds, decoded_labels = postprocess_text(decoded_preds,
decoded_labels)

    # ROUGE
    result_rouge = rouge.compute(predictions=decoded_preds,
references=decoded_labels, use_stemmer=True)
    # BLEU expects tokenized lists; evaluate's BLEU takes detokenized
strings and tokenizes internally
    try:
        result_bleu = bleu.compute(predictions=decoded_preds,
references=[[r] for r in decoded_labels])
    except Exception:
        # fallback: compute sacrebleu-like with single references
        result_bleu = {"bleu": 0.0}

    # BERTScore
    try:
        bs = bertscore.compute(predictions=decoded_preds,
references=decoded_labels, lang="en")
        result_berts = {
            "bertscore_precision": float(np.mean(bs["precision"])),
            "bertscore_recall": float(np.mean(bs["recall"])),
            "bertscore_f1": float(np.mean(bs["f1"])),
        }
    except Exception:
        result_berts = {"bertscore_f1": 0.0}

    # Exact match fraction
    exact_matches = float(np.mean([1.0 if p.strip() == r.strip() else
0.0 for p, r in zip(decoded_preds, decoded_labels)]))

    # length stats
    gen_len = np.mean([len(tokenizer.encode(p)) for p in
decoded_preds])

    metrics = {
        "rouge1": result_rouge["rouge1"],
        "rouge2": result_rouge["rouge2"],
        "rougeL": result_rouge["rougeL"],
        "bleu": result_bleu.get("bleu", 0.0),
        "exact_match": exact_matches,
        "gen_len": gen_len,
```

```
    }
    # add bertscore keys if available
    metrics.update(result_berts)
    # round metrics for readability
    metrics = {k: (float(v) if isinstance(v, (np.floating, float,
np.integer, int)) else v) for k, v in metrics.items()}
    return metrics


# ----------------------------
# Training arguments
# ----------------------------
training_args = TrainingArguments(
    output_dir=OUTPUT_DIR,
    eval_strategy="epoch",   # or "steps"
    save_strategy="epoch",
    learning_rate=LR,
    per_device_train_batch_size=TRAIN_BATCH_SIZE,
    per_device_eval_batch_size=EVAL_BATCH_SIZE,
    weight_decay=WEIGHT_DECAY,
    num_train_epochs=NUM_EPOCHS,
    logging_dir=os.path.join(OUTPUT_DIR, "logs"),
    logging_strategy="epoch",
    logging_steps=100,
    save_total_limit=3,
    fp16=torch.cuda.is_available(),
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    greater_is_better=False,
    seed=SEED,
    gradient_accumulation_steps=1,
)


# ----------------------------
# Initialize Trainer
# ----------------------------
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=None,
)


# ----------------------------
# Train
```

```python
# ----------------------------
print("Starting training...")
trainer.train()
print("Training finished.")

# Save the best model
print("Saving best model to", OUTPUT_DIR)
trainer.save_model(OUTPUT_DIR)
tokenizer.save_pretrained(OUTPUT_DIR)
```

```
/tmp/ipython-input-1771199625.py:29: FutureWarning: `tokenizer` is
deprecated and will be removed in version 5.0.0 for
`Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(

Starting training...

wandb: Currently logged in as: rk-rahul2903 (rk-rahul2903-maulana-
abul-kalam-azad-university-of-techn) to https://api.wandb.ai. Use
`wandb login --relogin` to force relogin

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Detected [huggingface_hub.inference, openai] in use.
wandb: Use W&B Weave for improved LLM call tracing. Install Weave with
`pip install weave` then add `import weave` to the top of your script.
wandb: For more information, check out the docs at: https://weave-
docs.wandb.ai/

<IPython.core.display.HTML object>

There were missing keys in the checkpoint model loaded:
['encoder.embed_tokens.weight', 'decoder.embed_tokens.weight',
'lm_head.weight'].

Training finished.
Saving best model to
/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints

('/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/
tokenizer_config.json',

'/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/special_tok
```

```
ens_map.json',

'/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/spiece.model',

'/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/added_tokens.json',

'/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/tokenizer.json')
```

Evaluate on test set with generation

```
print("Running generation on test set...")
def batch_generate(questions: List[str], batch_size=8, max_target_len=MAX_TARGET_LENGTH):
    generations = []
    model.eval()
    for i in range(0, len(questions), batch_size):
        batch_q = questions[i : i + batch_size]
        inputs = [TASK_PREFIX + q for q in batch_q]
        enc = tokenizer(
            inputs,
            max_length=MAX_SOURCE_LENGTH,
            truncation=True,
            padding="longest",
            return_tensors="pt",
        )
        input_ids = enc.input_ids.to(DEVICE)
        attention_mask = enc.attention_mask.to(DEVICE)

        with torch.no_grad():
            generated_ids = model.generate(
                input_ids=input_ids,
                attention_mask=attention_mask,
                max_length=max_target_len,
                num_beams=4,
                early_stopping=True,
                length_penalty=1.0,
                no_repeat_ngram_size=2,
            )

        decoded = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)
        generations.extend(decoded)
    return generations

test_questions = [r["question"] for r in test_pairs]
```

```
test_refs      = [r["summary"] for r in test_pairs]

preds = batch_generate(test_questions, batch_size=EVAL_BATCH_SIZE,
max_target_len=MAX_TARGET_LENGTH)

# Compute metrics using evaluate
print("Computing metrics on test set...")
# ROUGE
rouge_res = rouge.compute(predictions=preds, references=test_refs,
use_stemmer=True)
try:
    bleu_res = bleu.compute(predictions=preds, references=[[r] for r
in test_refs])
except Exception:
    bleu_res = {"bleu": 0.0}
# BERTScore
try:
    bs = bertscore.compute(predictions=preds, references=test_refs,
lang="en")
    bert_res = {
        "bertscore_precision": float(np.mean(bs["precision"])),
        "bertscore_recall": float(np.mean(bs["recall"])),
        "bertscore_f1": float(np.mean(bs["f1"])),
    }
except Exception:
    bert_res = {"bertscore_f1": 0.0}

exact_match = float(np.mean([1.0 if p.strip() == r.strip() else 0.0
for p, r in zip(preds, test_refs)]))
avg_gen_len = np.mean([len(tokenizer.encode(p)) for p in preds])

test_metrics = {
    "rouge1": rouge_res["rouge1"],
    "rouge2": rouge_res["rouge2"],
    "rougeL": rouge_res["rougeL"],
    "bleu": bleu_res.get("bleu", 0.0),
    "bertscore_f1": bert_res.get("bertscore_f1", 0.0),
    "exact_match": exact_match,
    "avg_gen_len": avg_gen_len,
}
print("Test metrics:")
for k, v in test_metrics.items():
    print(f"  {k}: {v:.4f}")


# Show example generation for one test question
# ------------------------------------------------
if len(test_questions) > 0:
    sample_idx = 0
    print("\nExample test item (index={}):".format(sample_idx))
```

```
    print("QUESTION:")
    print(test_questions[sample_idx])
    print("\nREFERENCE SUMMARY:")
    print(test_refs[sample_idx])
    print("\nMODEL PREDICTION:")
    print(preds[sample_idx])
else:
    print("No test examples found.")

# Save predictions to disk (optional)
out_preds_file = os.path.join(OUTPUT_DIR, "test_predictions.jsonl")
with open(out_preds_file, "w", encoding="utf-8") as f:
    for i, (q, ref, pred) in enumerate(zip(test_questions, test_refs,
preds)):
        f.write(json.dumps({"_id": test_pairs[i].get("_id"),
"question": q, "reference": ref, "prediction": pred}) + "\n")
print("Saved test predictions to", out_preds_file)
```

```
Running generation on test set...
Computing metrics on test set...

Some weights of RobertaModel were not initialized from the model
checkpoint at roberta-large and are newly initialized:
['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.

Test metrics:
  rouge1: 0.1861
  rouge2: 0.0490
  rougeL: 0.1374
  bleu: 0.0008
  bertscore_f1: 0.8461
  exact_match: 0.0000
  avg_gen_len: 36.5000

Example test item (index=0):
QUESTION:
how to prevent conjunctivitis

REFERENCE SUMMARY:
Good hygiene can help prevent the spread of conjunctivitis. Things you
can do include change your pillowcases often, do not share eye makeup,
do not share towels or handkerchiefs, handle your contact lenses
properly, keep your hands away from the eye, and wash your hands
often.

MODEL PREDICTION:
Conjunctivitis is a condition that affects the skin, the eyes, and
other parts of the body. It can be treated with antibiotics,
```

```
antifungals, or other medications.
Saved test predictions to
/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/test_predict
ions.jsonl
```

TASK 2

```
def load_json_records(path):
    """Load list or dict JSON into list of dicts."""
    with open(path, "r", encoding="utf-8") as f:
        data = json.load(f)
    if isinstance(data, dict):
        return list(data.values())
    elif isinstance(data, list):
        return data
    else:
        raise ValueError(f"Unsupported JSON format: {type(data)}")

# ----------------------------------------
# Build input-output pairs (Task 2 version)
# ----------------------------------------
def build_task2_pairs(records, question_field="question",
answers_field="answers", summary_field="multi_abs_summ"):
    pairs = []
    for r in records:
        q = r.get(question_field)
        answers = r.get(answers_field)
        summ = r.get(summary_field)
        if not q or not answers or not summ:
            continue
        # If answers is a dict (as in your data)
        if isinstance(answers, dict):
            # Pull the abstractive summary from each answer if it
exists
            answer_texts = [a.get("answer_abs_summ", "").strip() for a
in answers.values() if a.get("answer_abs_summ")]
        elif isinstance(answers, list):
            answer_texts = [a.strip() for a in answers if
isinstance(a, str)]
        else:
            answer_texts = [str(answers).strip()]
        if not answer_texts:
            continue
```

```
        joined_answers = " [SEP] ".join(answer_texts)
        inp = f"multianswer: question: {q.strip()} \nContext:
{joined_answers}"
        pairs.append({"input_text": inp, "target_text": summ.strip()})
    return pairs


# ------------------------------
# Load datasets
# ------------------------------
print("Loading MEDIQA-AnS datasets...")
train_recs = load_json_records(TRAIN_FILE)
val_recs   = load_json_records(VALID_FILE)
test_recs  = load_json_records(TEST_FILE)

train_pairs = build_task2_pairs(train_recs)
val_pairs   = build_task2_pairs(val_recs)
test_pairs  = build_task2_pairs(test_recs)

print(f"Data sizes - train: {len(train_pairs)}, val: {len(val_pairs)},
test: {len(test_pairs)}")

dataset = DatasetDict({
    "train": Dataset.from_list(train_pairs),
    "validation": Dataset.from_list(val_pairs),
    "test": Dataset.from_list(test_pairs),
})
```

```
Loading MEDIQA-AnS datasets...
Data sizes - train: 110, val: 16, test: 30
```

```
# ------------------------------
# Tokenizer & model
# ------------------------------
tokenizer = T5TokenizerFast.from_pretrained(MODEL_NAME)
model =
T5ForConditionalGeneration.from_pretrained(MODEL_NAME).to(DEVICE)

def preprocess(examples):
    model_inputs = tokenizer(
        examples["input_text"],
        max_length=MAX_SOURCE_LENGTH,
        padding="max_length",
        truncation=True,
    )
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
```

```
            examples["target_text"],
            max_length=MAX_TARGET_LENGTH,
            padding="max_length",
            truncation=True,
        )
    # Mask out padding tokens
    labels["input_ids"] = [
        [(token if token != tokenizer.pad_token_id else -100) for
token in label]
        for label in labels["input_ids"]
    ]
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

print("Tokenizing data...")
tokenized = dataset.map(preprocess, batched=True,
remove_columns=dataset["train"].column_names)

# ----------------------------
# Training arguments
# ----------------------------
training_args = TrainingArguments(
    output_dir=OUTPUT_DIR,
    eval_strategy="epoch",
    save_strategy="epoch",
    logging_strategy="epoch",
    learning_rate=LR,
    per_device_train_batch_size=TRAIN_BATCH_SIZE,
    per_device_eval_batch_size=EVAL_BATCH_SIZE,
    num_train_epochs=NUM_EPOCHS,
    weight_decay=0.01,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    greater_is_better=False,
    fp16=torch.cuda.is_available(),
    save_total_limit=3,
    seed=SEED,
)

data_collator = DataCollatorForSeq2Seq(tokenizer, model)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized["train"],
    eval_dataset=tokenized["validation"],
    tokenizer=tokenizer,
    data_collator=data_collator,
)
```

```
print("Starting fine-tuning...")
trainer.train()
print("Training complete; saving best model...")
trainer.save_model(OUTPUT_DIR)
tokenizer.save_pretrained(OUTPUT_DIR)
```

```
Tokenizing data...
```


```
{"model_id":"be7f0169eeb24f308fb72a3330ed2374","version_major":2,"vers
ion_minor":0}

{"model_id":"4e7c610eabbc4d1f9a23330257b8defb","version_major":2,"vers
ion_minor":0}

{"model_id":"88d98058d6ca4d19b21286b48e18dc4a","version_major":2,"vers
ion_minor":0}
```


```
/tmp/ipython-input-867705401.py:55: FutureWarning: `tokenizer` is
deprecated and will be removed in version 5.0.0 for
`Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(

Starting fine-tuning...

<IPython.core.display.HTML object>

There were missing keys in the checkpoint model loaded:
['encoder.embed_tokens.weight', 'decoder.embed_tokens.weight',
'lm_head.weight'].

Training complete; saving best model...

('/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/
tokenizer_config.json',

'/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/special_tok
ens_map.json',

'/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/spiece.mode
l',

'/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/added_token
s.json',

'/content/drive/MyDrive/DeepLearning/t5_meqsum_checkpoints/tokenizer.j
son')
```

```
# ----------------------------
# Evaluation on test set
# ----------------------------
rouge = evaluate.load("rouge")
```

```
def generate_batch_texts(batch):
    inputs = tokenizer(batch["input_text"], return_tensors="pt",
                       padding="longest", truncation=True,
                       max_length=MAX_SOURCE_LENGTH).to(DEVICE)
    output_ids = model.generate(**inputs,
max_length=MAX_TARGET_LENGTH, num_beams=4)
    batch["predicted_summary"] = tokenizer.batch_decode(output_ids,
skip_special_tokens=True)
    return batch

print("Generating summaries on test set...")
preds_dataset = dataset["test"].map(generate_batch_texts,
batched=True, batch_size=EVAL_BATCH_SIZE)
results =
rouge.compute(predictions=preds_dataset["predicted_summary"],
                        references=preds_dataset["target_text"],
use_stemmer=True)
print("ROUGE results:", results)

# Example output
if len(preds_dataset) > 0:
    i = 0
    print("\nExample prediction: ")
    print("QUESTION: ", test_pairs[i]["input_text"][:5000], "...")
    print("\n")
    print("REFERENCE SUMMARY: ", test_pairs[i]["target_text"])
    print("MODEL SUMMARY: ", preds_dataset[i]["predicted_summary"])
```

```
Generating summaries on test set...
```

```
Parameter 'function'=<function generate_batch_texts at 0x7a670a83f060>
of the transform datasets.arrow_dataset.Dataset._map_single couldn't
be hashed properly, a random hash was used instead. Make sure your
transforms and parameters are serializable with pickle or dill for the
dataset fingerprinting and caching to work. If you reuse this
transform, the caching mechanism will consider it to be different from
the previous calls and recompute everything. This warning is only
shown once. Subsequent hashing failures won't be shown.
WARNING:datasets.fingerprint:Parameter 'function'=<function
generate_batch_texts at 0x7a670a83f060> of the transform
datasets.arrow_dataset.Dataset._map_single couldn't be hashed
properly, a random hash was used instead. Make sure your transforms
and parameters are serializable with pickle or dill for the dataset
fingerprinting and caching to work. If you reuse this transform, the
caching mechanism will consider it to be different from the previous
calls and recompute everything. This warning is only shown once.
Subsequent hashing failures won't be shown.
```

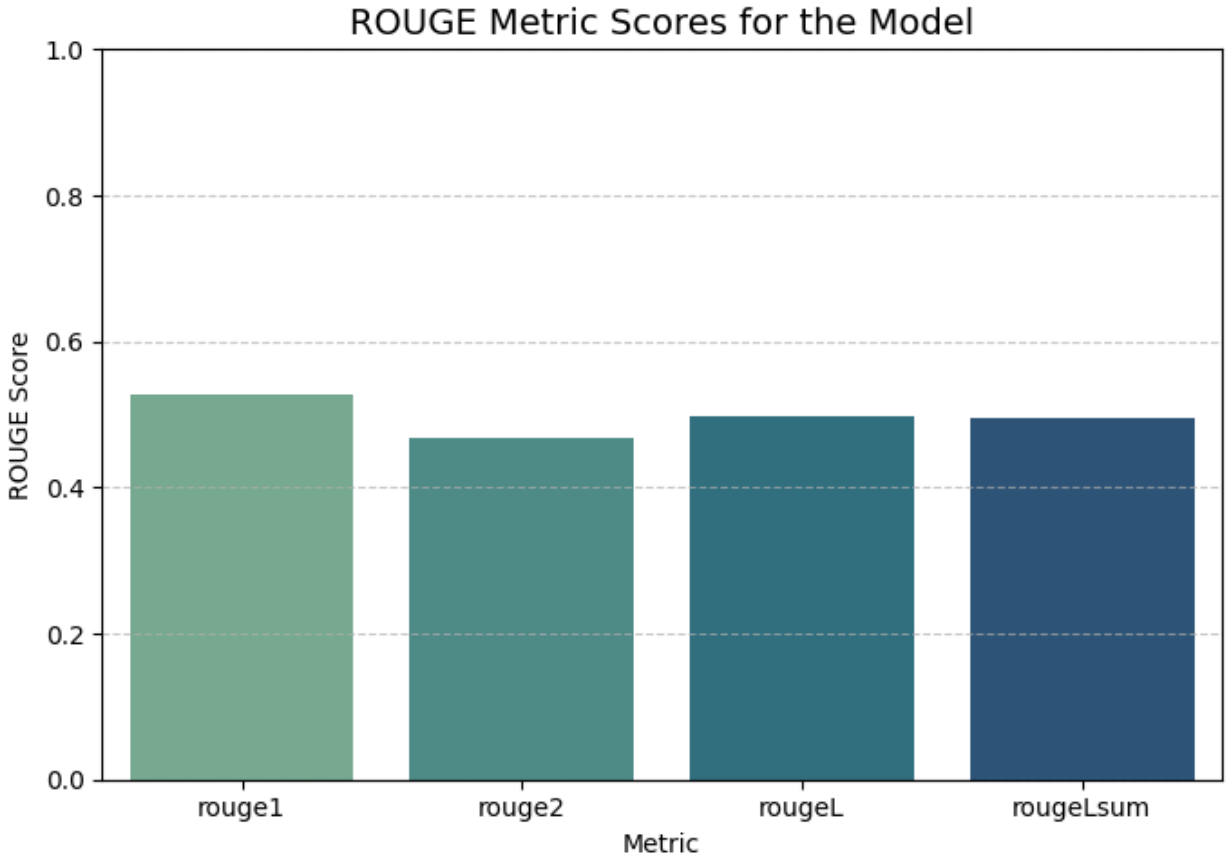
{"model_id":"51224c2f4c65463da8f2009c15b8a107","version_major":2,"version_minor":0}

```
ROUGE results: {'rouge1': np.float64(0.5266620737904288), 'rouge2':
np.float64(0.4678805795199292), 'rougeL':
np.float64(0.4988204035695265), 'rougeLsum':
np.float64(0.49477271627395336)}

Example prediction:
QUESTION:  multianswer: question: how to prevent conjunctivitis
Context: To help prevent newborn conjunctivitis, pregnant women should
get treatment for diseases spread through sexual contact. Putting eye
drops into all infants' eyes in the delivery room after birth can also
help prevent infections. [SEP] Using an air conditioner or moving to a
cooler climate may prevent vernal conjunctivitis from getting worse in
the future. [SEP] Good hygiene can help prevent the spread of
conjunctivitis. Things you can do include change your pillowcases
often, do not share eye makeup, do not share towels or handkerchiefs,
handle your contact lenses properly, keep your hands away from the
eye, and wash your hands often. [SEP] Conjunctivitis is swelling or
infection of the membrane that lines the eyelids and covers the white
part of the eye. Quick diagnosis and treatment usually leads to good
outcomes. ...


REFERENCE SUMMARY:  Good hygiene can help prevent the spread of
conjunctivitis. Things you can do include change your pillowcases
often, do not share eye makeup, do not share towels or handkerchiefs,
handle your contact lenses properly, keep your hands away from the
eye, and wash your hands often.
MODEL SUMMARY:  To help prevent newborn conjunctivitis, pregnant women
should get treatment for diseases spread through sexual contact.
Putting eye drops into all infants' eyes in the delivery room after
birth can also help prevent infections. Using an air conditioner or
moving to a cooler climate may prevent vernal con
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

df = pd.DataFrame(list(results.items()), columns=["Metric", "Score"])

# Plot
plt.figure(figsize=(7, 5))
sns.barplot(x="Metric", y="Score", data=df, palette="crest")
plt.title("ROUGE Metric Scores for the Model", fontsize=14)
plt.ylabel("ROUGE Score")
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-4060856738.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x="Metric", y="Score", data=df, palette="crest")
```



login to huggingface

```
from huggingface_hub import notebook_login

notebook_login()
```

{"model_id":"28626d899eb24ae583210470132bdf04","version_major":2,"version_minor":0}

```
# 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

HF_REPO_ID = "rk2903/t5_meqsum_summarizer"

model.push_to_hub(repo_id=HF_REPO_ID)
tokenizer.push_to_hub(repo_id=HF_REPO_ID)
print(f"Model and tokenizer pushed to Hugging Face Hub at:
{HF_REPO_ID}")
```

{"model_id":"0aef779724694b1abe48557105d92e12","version_major":2,"version_minor":0}

{"model_id":"0fee7734f4c64388b6ec4f5b94b22b79","version_major":2,"version_minor":0}

{"model_id":"9cd3ce3906c0428c839b53f8aa2431eb","version_major":2,"version_minor":0}

```
No files have been modified since last commit. Skipping to prevent
empty commit.
WARNING:huggingface_hub.hf_api:No files have been modified since last
commit. Skipping to prevent empty commit.
```

{"model_id":"19cd9c4f51a249c58db029fbe43cf99e","version_major":2,"version_minor":0}

{"model_id":"7bdc688ba3bc43a198e5297dd40f9b79","version_major":2,"version_minor":0}

{"model_id":"690776b7227d4688a45ebf6ef64ca6dd","version_major":2,"version_minor":0}

```
No files have been modified since last commit. Skipping to prevent
empty commit.
WARNING:huggingface_hub.hf_api:No files have been modified since last
commit. Skipping to prevent empty commit.

Model and tokenizer pushed to Hugging Face Hub at:
rk2903/t5_meqsum_summarizer
```

```
print("Model and tokenizer successfully pushed to Hugging Face Hub.")
```

```
Model and tokenizer successfully pushed to Hugging Face Hub.
```

push_dataset_to_hub

```
from datasets import DatasetDict
```

```
HF_TRAIN_DATASET_REPO_ID = "rk2903/t5-meqsum-train-dataset"
HF_VALID_DATASET_REPO_ID = "rk2903/t5-meqsum-validation-dataset"
HF_TEST_DATASET_REPO_ID = "rk2903/t5-meqsum-test-dataset"

dataset.push_to_hub(repo_id=HF_TRAIN_DATASET_REPO_ID)
dataset.push_to_hub(repo_id=HF_VALID_DATASET_REPO_ID)
dataset.push_to_hub(repo_id=HF_TEST_DATASET_REPO_ID)

print(f"Dataset successfully pushed to Hugging Face Hub at:
{HF_TRAIN_DATASET_REPO_ID}")
print(f"Dataset successfully pushed to Hugging Face Hub at:
{HF_VALID_DATASET_REPO_ID}")
print(f"Dataset successfully pushed to Hugging Face Hub at:
{HF_TEST_DATASET_REPO_ID}")
```


```
{"model_id":"c865ec9b8afb4e4ca2c2621bf264fed2","version_major":2,"vers
ion_minor":0}

{"model_id":"cf9043925b204f86992848c9317bc071","version_major":2,"vers
ion_minor":0}

{"model_id":"e838ed4936ec4115959ef7d1ebfb25b6","version_major":2,"vers
ion_minor":0}

{"model_id":"7a23e4ca05ef4453a5bff8ecc1608f7a","version_major":2,"vers
ion_minor":0}

{"model_id":"cd538647ede34c2aa0a41e05b783925e","version_major":2,"vers
ion_minor":0}

{"model_id":"89b41a20308b4473b928427c517c7e3d","version_major":2,"vers
ion_minor":0}

{"model_id":"1cae5ed7f7a846648c734252a3ce6341","version_major":2,"vers
ion_minor":0}

{"model_id":"47c1ae77179e4592bba30b9d8953e3c8","version_major":2,"vers
ion_minor":0}

{"model_id":"a44d10d0b9e9408baf585c3a5c57e439","version_major":2,"vers
ion_minor":0}

{"model_id":"f3a513f95e8048599abac1c2d4c7628d","version_major":2,"vers
ion_minor":0}

{"model_id":"222a757fd4024a78939ac5ee2ca4690a","version_major":2,"vers
ion_minor":0}

{"model_id":"e450280bc8024bd6af25d1c45634d912","version_major":2,"vers
ion_minor":0}
```

{"model_id":"f3c10021b87c49e697f7908966134f2f","version_major":2,"version_minor":0}

{"model_id":"4f72860be7b842c484dda22906d6df2f","version_major":2,"version_minor":0}

{"model_id":"1cbad08d5ab64eddb5812a6ba542f7d1","version_major":2,"version_minor":0}

{"model_id":"5f3d5a9d29fa48b4b79ccb124e891300","version_major":2,"version_minor":0}

{"model_id":"540375eb74cc4acd94989493b794f061","version_major":2,"version_minor":0}

{"model_id":"650ac975fa4e4141a65c1466fed0907c","version_major":2,"version_minor":0}

{"model_id":"ccb8ab01ae164906ba34427092acb972","version_major":2,"version_minor":0}

{"model_id":"4aff5149ab414fdcb3e32f6831828bed","version_major":2,"version_minor":0}

{"model_id":"3db9301b3b434326894b92abf475d072","version_major":2,"version_minor":0}

{"model_id":"28b6e19b27d54a35ae7e4c656d1f996c","version_major":2,"version_minor":0}

{"model_id":"6042322547bc463498c7d705561e8093","version_major":2,"version_minor":0}

{"model_id":"633cbeb02f7746be9e63b5ae4a15e733","version_major":2,"version_minor":0}

{"model_id":"2c256fc1c72a4c8cab5acc21157521df","version_major":2,"version_minor":0}

{"model_id":"4b40694d76c64610a79662ab970fdb85","version_major":2,"version_minor":0}

{"model_id":"4661582071664353bc756f672586f3e7","version_major":2,"version_minor":0}

{"model_id":"3f8f402af6ef45febeb10ec5d7594e06","version_major":2,"version_minor":0}

{"model_id":"7cf78a33a46344fcb7afd4bfd0b92948","version_major":2,"version_minor":0}

{"model_id":"d869a948a6d142f0846e6082e9c89f0c","version_major":2,"version_minor":0}

{"model_id":"702516f27be34aef85dafacd86936084","version_major":2,"version_minor":0}

{"model_id":"f174bfd610b148afb1c4e9e14e87f582","version_major":2,"version_minor":0}

{"model_id":"663d3225fde04feaafaa10968e5b74ab","version_major":2,"version_minor":0}

{"model_id":"58ec249806bf4481841614905da386a1","version_major":2,"version_minor":0}

{"model_id":"d2a45e4f52844f30b200a507ba85f8b0","version_major":2,"version_minor":0}

{"model_id":"e864e4b024884a3388258dcdfb99a5f8","version_major":2,"version_minor":0}

{"model_id":"7ac6dfc35c2f4dbabcdbe90bf3cc4cab","version_major":2,"version_minor":0}

{"model_id":"be700feecffb4cf1a1bec007fa14ce98","version_major":2,"version_minor":0}

{"model_id":"4e486dec9ad540b3ae26b2a7a85c4339","version_major":2,"version_minor":0}

{"model_id":"bae684f6d51d41799b033e0d473134ac","version_major":2,"version_minor":0}

{"model_id":"4934a846b5994048a6d75a06b5c22c51","version_major":2,"version_minor":0}

{"model_id":"ad7829f1db094999804583bcaf32d57d","version_major":2,"version_minor":0}

{"model_id":"39a3ad4e2aa943f9b3727d8cd66f8f12","version_major":2,"version_minor":0}

{"model_id":"abfeeff8f6bd4499924d745e8cf1a79c","version_major":2,"version_minor":0}

{"model_id":"beb585be271b4015a1941f9fc20c4f2b","version_major":2,"version_minor":0}

```
Dataset successfully pushed to Hugging Face Hub at: rk2903/t5-meqsum-train-dataset
Dataset successfully pushed to Hugging Face Hub at: rk2903/t5-meqsum-validation-dataset
Dataset successfully pushed to Hugging Face Hub at: rk2903/t5-meqsum-test-dataset
```