Personalized Healthcare QA System (Chatbot) - V3 - PubMedQA

Ed:215279167

Objective:

Develop a healthcare chatbot that answers patient inquiries on medications, symptoms, and treatments while reducing hallucinations.

Update and changes:

This updated notebook reflects changes learned from weeks 1 -7 experiments with fine-tuning. Main changes include:

- Use single dataset of PubMedQA. Initial attempts to use a combination of MedDialog and PubMedQA didn't provide significant improvements in evaluation, and beleive this is due to differing data quality in MedDialog dataset.
- Significant increase of data used for fine-tuning

Datasets:

- PubMedQA – Biomedical QA dataset - https://huggingface.co/datasets/qiaojin/PubMedQA

Task Breakdown:

1. Train models on medical QA datasets.
2. Fine-tune for patient-friendly responses (simplified, clear language).
3. Ensure context-aware, regulatory-compliant answers:
   o Implement FDA/TGA guideline alignment. o Develop a retrieval-based validation for generated answers.
4. Deploy as a chatbot interface (React-based UI + API integration).
5. Implement real-time fact-checking:
   o Confidence score visualization (green = high confidence, yellow = medium, red = low confidence). o Integration with PubMed and trusted medical sources.

Models to Use:

• Llama-2 7B

Evaluation Metrics:

- Rouge, BLEU, Meteor, BertScore, Manual inspection

---

1. Import libraries and model to prepare for fine tuning

```
!pip install datasets requests bitsandbytes accelerate peft trl sentencepiece wandb transformers evaluate rouge_score bert-score
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 3.5 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 53.2 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 42.7 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 44.2 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.2 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.4 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 13.3 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.5 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.4 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 81.2 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 16.9 MB/s eta 0:00:00
Building wheels for collected packages: rouge_score
  Building wheel for rouge_score (setup.py) ... done
  Created wheel for rouge_score: filename=rouge_score-0.1.2-py3-none-any.whl size=24934 sha256=61018e05c29de1ac53e312cf59c6999d77
  Stored in directory: /root/.cache/pip/wheels/1e/19/43/8a442dc83660ca25e163e1bd1f89919284ab0d0c1475475148
Successfully built rouge_score
```

IMPORTANT: Restart Colab runtime after PIP install!!!

2. Implement base model and imports

First step is to import all necessary libraries and implement base model, to validate it can generate a response from prompt.

```
from google.colab import userdata
from huggingface_hub import login
from transformers import (
    AutoTokenizer,
    AutoModelForCausalLM,
    BitsAndBytesConfig,
    TrainingArguments,
    logging,
    pipeline
)
from peft import LoraConfig, PeftModel, prepare_model_for_kbit_training, get_peft_model, TaskType
from trl import SFTTrainer, SFTConfig, DataCollatorForCompletionOnlyLM
from datasets import load_dataset, Dataset, DatasetDict
from datetime import datetime
import torch
import wandb
import pandas as pd
import os
import random
import re
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
import evaluate
import numpy as np
from bert_score import BERTScorer
```

```
model_name = "meta-llama/Llama-2-7b-chat-hf"
project_name = "ed_medical"
hf_username = "digitalblue"
```

```
# log into hugging face and wandb
hf_token = userdata.get('HF_TOKEN')
login(hf_token)

wandb_api_key = userdata.get('WANDB_API_KEY')
os.environ["WANDB_API_KEY"] = wandb_api_key
wandb.login()

# Configure Weights & Biases to record against our project
os.environ["WANDB_PROJECT"] = "ed_medical"
os.environ["WANDB_LOG_MODEL"] = "checkpoint" if True else "end"
os.environ["WANDB_WATCH"] = "gradients"
```

```
wandb: Currently logged in as: digitalblue (digitalblue-ai) to https://api.wandb.ai. Use `wandb login --relogin` to force relogin
```

```
# quantisation config to use less memory when loading model
quant_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.bfloat16,
    bnb_4bit_quant_type="nf4"
)
```

```
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="auto",
    quantization_config=quant_config)
```

```
config.json: 100%  614/614 [00:00<00:00, 54.9kB/s]
model.safetensors.index.json: 100%  26.8k/26.8k [00:00<00:00, 1.98MB/s]
Fetching 2 files: 100%  2/2 [01:16<00:00, 76.76s/it]
model-00001-of-00002.safetensors: 100%  9.98G/9.98G [01:16<00:00, 304MB/s]
model-00002-of-00002.safetensors: 100%  3.50G/3.50G [00:38<00:00, 110MB/s]
Loading checkpoint shards: 100%  2/2 [01:00<00:00, 27.73s/it]
generation_config.json: 100%  188/188 [00:00<00:00, 11.1kB/s]
```

```
memory = model.get_memory_footprint() / 1e6
print(f"Memory footprint: {memory:,.1f} MB")
```

```
Memory footprint: 3,762.8 MB
```

```
# model architecture
model
```

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(32000, 4096)
    (layers): ModuleList(
      (0-31): 32 x LlamaDecoderLayer(
        (self_attn): LlamaAttention(
          (q_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (v_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (o_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear4bit(in_features=4096, out_features=11008, bias=False)
          (up_proj): Linear4bit(in_features=4096, out_features=11008, bias=False)
          (down_proj): Linear4bit(in_features=11008, out_features=4096, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm((4096,), eps=1e-05)
        (post_attention_layernorm): LlamaRMSNorm((4096,), eps=1e-05)
      )
    )
    (norm): LlamaRMSNorm((4096,), eps=1e-05)
    (rotary_emb): LlamaRotaryEmbedding()
  )
  (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)
```

```
# initialise tokenizer and pipeline
tokenizer = AutoTokenizer.from_pretrained(model_name, token=hf_token)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
tokenizer_config.json: 100%  1.62k/1.62k [00:00<00:00, 89.5kB/s]
tokenizer.model: 100%  500k/500k [00:00<00:00, 7.57MB/s]
tokenizer.json: 100%  1.84M/1.84M [00:00<00:00, 5.63MB/s]
special_tokens_map.json: 100%  414/414 [00:00<00:00, 28.9kB/s]
```

```
messages = [
    {"role": "system", "content": "You are a helpful personalised medical assistant"},
    {"role": "user", "content": "How can I get rid of the flu?"}
 ]
```

```
# verify base model generates response
inputs = tokenizer.apply_chat_template(messages, return_tensors="pt").to("cuda")
```

```
outputs = model.generate(inputs, max_new_tokens=250)
print(tokenizer.decode(outputs[0]))
```

```
<s> [INST] <<SYS>>
You are a helpful personalised medical assistant
<</SYS>>

How can I get rid of the flu? [/INST]  Hello there! I'm glad you asked! 😊 As a helpful personalized medical assistant, I'm here to

First and foremost, it's important to understand that the flu is a viral infection, and there's no magic pill or potion that can mak

1. Stay hydrated: Drink plenty of fluids, such as water, tea, or soup, to help loosen congestion and replace lost fluids. Aim for at
2. Rest: Give your body the rest it needs to fight off the infection. Try to get at least 8 hours of sleep each night and take naps
3. Use over-the-counter medications: Over-the-counter medications like acetaminophen (Tylenol) or ibuprofen (Advil, Motrin) can help
```

3. Dataset analysis and preprocessing for Llama

The datasets need to be analysed and prepared for fine-tuning the model. This includes removing invalid data and formatting into the required format with special tokens the Llama 2 model requires. Dataset acquired from HuggingFace

- PubMedQA - https://huggingface.co/datasets/qiaojin/PubMedQA

```
# format input from datasets into prompt format required by llama model
def format_llama_prompt(user_message, model_answer):
  prompt = '<s>[INST] ' # special token - commence instruct
  prompt += user_message.strip()
  prompt += ' [/INST] ' # special token - end instruct
  prompt += model_answer.strip()
  prompt += ' </s>'# special token - end
  return prompt

def format_llama_prompt_with_context(user_message, model_answer, context):
  prompt = '<s>[INST] Given this context: ' # special token - commence instruct
  prompt += context['contexts'][0].strip()
  prompt += ' Question: '
  prompt += user_message.strip()
  prompt += ' [/INST] ' # special token - end instruct
  prompt += model_answer.strip()
  prompt += ' </s>'# special token - end
  return prompt
```

```
# load pub_med_qa dataset
#pub_med_qa = pd.read_parquet("hf://datasets/qiaojin/PubMedQA/pqa_unlabeled/train-00000-of-00001.parquet")
pub_med_qa = load_dataset("qiaojin/PubMedQA", "pqa_artificial")
```

```
README.md: 100%  5.19k/5.19k [00:00<00:00, 275kB/s]
train-00000-of-00001.parquet: 100%  233M/233M [00:01<00:00, 226MB/s]
Generating train split: 100%  211269/211269 [00:03<00:00, 52551.39 examples/s]
```

```
pub_med_qa
```

```
DatasetDict({
    train: Dataset({
        features: ['pubid', 'question', 'context', 'long_answer', 'final_decision'],
        num_rows: 211269
    })
})
```

```
pub_med_qa['train'][0]
```

```
{'pubid': 25429730,
 'question': 'Are group 2 innate lymphoid cells ( ILC2s ) increased in chronic rhinosinusitis with nasal polyps or eosinophilia?',
 'context': {'contexts': ['Chronic rhinosinusitis (CRS) is a heterogeneous disease with an uncertain pathogenesis. Group 2 innate lymphoid cells (ILC2s) represent a recently discovered cell population which has been implicated in driving Th2 inflammation in CRS; however, their relationship with clinical disease characteristics has yet to be investigated.',
   'The aim of this study was to identify ILC2s in sinus mucosa in patients with CRS and controls and compare ILC2s across characteristics of disease.',
   'A cross-sectional study of patients with CRS undergoing endoscopic sinus surgery was conducted. Sinus mucosal biopsies were obtained during surgery and control tissue from patients undergoing pituitary tumour resection through transphenoidal approach. ILC2s were identified as CD45(+) Lin(-) CD127(+) CD4(-) CD8(-) CRTH2(CD294)(+) CD161(+) cells in single cell suspensions through flow cytometry. ILC2 frequencies, measured as a percentage of CD45(+) cells, were compared across CRS phenotype, endotype, inflammatory CRS subtype and other disease characteristics including blood eosinophils, serum IgE, asthma status and nasal symptom score.',
   '35 patients (40% female, age 48 ± 17 years) including 13 with eosinophilic CRS (eCRS), 13 with non-eCRS and 9 controls were recruited. ILC2 frequencies were associated with the presence of nasal polyps (P = 0.002) as well as high tissue eosinophilia (P = 0.004) and eosinophil-dominant CRS (P = 0.001) (Mann-Whitney U). They were also associated with increased blood eosinophilia (P = 0.005). There were no significant associations found between ILC2s and serum total IgE and allergic disease. In the CRS with nasal polyps (CRSwNP) population, ILC2s were increased in patients with co-existing asthma (P = 0.03). ILC2s were also correlated with
```

```
    worsening nasal symptom score in CRS (P = 0.04).'],
      'labels': ['BACKGROUND', 'OBJECTIVE', 'METHODS', 'RESULTS'],
      'meshes': ['Adult',
       'Aged',
       'Antigens, Surface',
       'Case-Control Studies',
       'Chronic Disease',
       'Eosinophilia',
       'Female',
       'Humans',
       'Hypersensitivity',
       'Immunity, Innate',
       'Immunoglobulin E',
       'Immunophenotyping',
       'Leukocyte Count',
       'Lymphocyte Subsets',
       'Male',
       'Middle Aged',
       'Nasal Mucosa',
       'Nasal Polyps',
       'Neutrophil Infiltration',
       'Patient Outcome Assessment',
       'Rhinitis',
       'Sinusitis',
       'Young Adult']},
     'long_answer': 'As ILC2s are elevated in patients with CRSwNP, they may drive nasal polyp formation in CRS. ILC2s are also linked
    with high tissue and blood eosinophilia and have a potential role in the activation and survival of eosinophils during the Th2
    immune response. The association of innate lymphoid cells in CRS provides insights into its pathogenesis.',
     'final_decision': 'yes'}
```

```
# create list of formatted prompts
pub_med_qa_as_prompt = []
for item in pub_med_qa['train']:
  if item['final_decision'] == 'yes': # only get rows with yes decision
    #pub_med_qa_as_prompt.append(format_llama_prompt(item['question'], item['long_answer']))
    pub_med_qa_as_prompt.append(format_llama_prompt_with_context(item['question'], item['long_answer'], item['context']))

```

```
print(len(pub_med_qa_as_prompt))
print(pub_med_qa_as_prompt[0])
```

```
196144
<s>[INST] Given this context: Chronic rhinosinusitis (CRS) is a heterogeneous disease with an uncertain pathogenesis. Group 2 innate
```

```
pub_med_qa_as_prompt[200]
```

```
'<s>[INST] Given this context: To investigate the effect of the percentage of free prostate-specific antigen (%fPSA) on future pros
tate cancer risk. Question: Is low percentage of free prostate-specific antigen ( PSA ) a strong predictor of later detection of pr
ostate cancer among Japanese men with serum levels of total PSA of 4.0 ng/mL or less? [/INST] A low %fPSA is a strong predictor of
```

```
# collect all formatted data into one dataset
prompt_dataset = pub_med_qa_as_prompt # + med_dialog_hcm_as_prompt + med_dialog_ic_as_prompt
print(len(prompt_dataset))
```

```
196144
```

```
# shuffle the combined dataset to disperse the data
# random.seed(42)
# random.shuffle(prompt_dataset)
```

```
len(prompt_dataset)
```

```
196144
```

```
# convert to huggingface dataset, create splits and push to hub
prompt_hf_dataset = Dataset.from_dict({"text": prompt_dataset})
```

```
ds_train = prompt_hf_dataset.train_test_split(test_size=0.2, seed=42)
```

```
ds_train
```

```
DatasetDict({
    train: Dataset({
        features: ['text'],
        num_rows: 156915
    })
    test: Dataset({
        features: ['text'],
        num_rows: 39229
```

```
        })
    })
```

```
ds_test = ds_train['test'].train_test_split(test_size=0.5, seed=42)
ds_test
```

```
DatasetDict({
    train: Dataset({
        features: ['text'],
        num_rows: 19614
    })
    test: Dataset({
        features: ['text'],
        num_rows: 19615
    })
})
```

```
ds_splits = DatasetDict({
    'train': ds_train['train'],
    'validation': ds_test['train'],
    'test': ds_test['test']
})
```

```
ds_splits
```

```
DatasetDict({
    train: Dataset({
        features: ['text'],
        num_rows: 156915
    })
    validation: Dataset({
        features: ['text'],
        num_rows: 19614
    })
    test: Dataset({
        features: ['text'],
        num_rows: 19615
    })
})
```

```
# push to hugging face
ds_splits.push_to_hub(f"{hf_username}/{project_name}-pubmedqa-artifical-with-context", private=True)
```

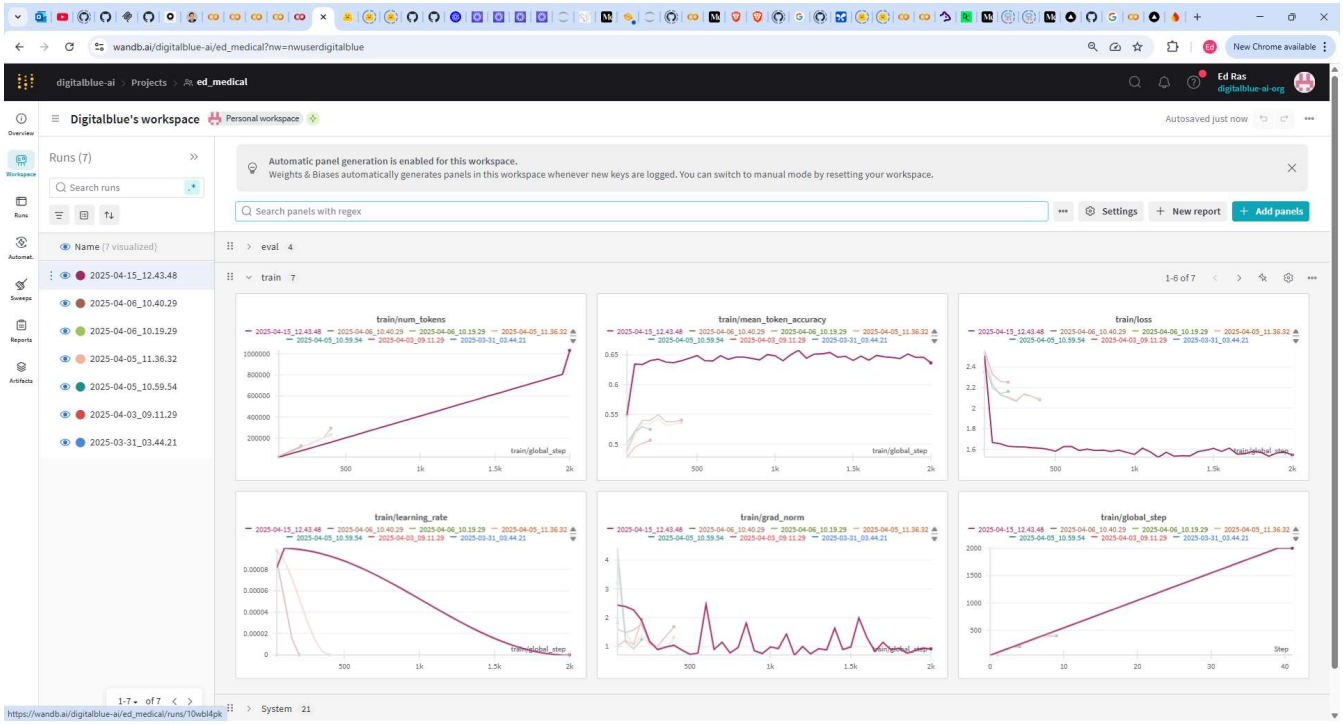
```
Uploading the dataset shards: 100%  1/1 [00:09<00:00,  9.05s/it]
Creating parquet from Arrow format: 100%  157/157 [00:04<00:00, 27.06ba/s]
Uploading the dataset shards: 100%  1/1 [00:02<00:00,  2.72s/it]
Creating parquet from Arrow format: 100%  20/20 [00:01<00:00, 22.05ba/s]
Uploading the dataset shards: 100%  1/1 [00:01<00:00,  1.29s/it]
Creating parquet from Arrow format: 100%  20/20 [00:00<00:00, 37.79ba/s]
CommitInfo(commit_url='https://huggingface.co/datasets/digitalblue/ed_medical-pubmedqa-artifical-with-context/commit/d305407ce4ca80e249fea443677aac7451183561', commit_message='Upload dataset', commit_description='', oid='d305407ce4ca80e249fea443677aac7451183561', pr_url=None,
```

4. Model Training Pipeline

The model training pipeline is setup for fine-tuning. During this process additional data processing was required to filter out data with execessive token length that would exhaust GPU resources.

Initially four variants(A,B,C,D) of the model were fine-tuned on combination dataset, however focus is now on model variant E. It maintains integration with HuggingFace and Weight and Bias platform so models and runs are saved for later retrieval.

The models were fine-tuned on the **train** set, and evaluated with **validation** set.




```
# load pre-processed dataset created in last step from hugging face
#qa_dataset = load_dataset("digitalblue/ed_medical")
#qa_dataset = load_dataset("digitalblue/ed_medical-pubmedqa-artifical") # load only pubmedqa YES rows
qa_dataset = load_dataset("digitalblue/ed_medical-pubmedqa-artifical-with-context") # load only pubmedqa YES rows
qa_dataset
```

```
README.md: 100%  532/532 [00:00<00:00, 35.4kB/s]
train-00000-of-00001.parquet: 100%  63.8M/63.8M [00:01<00:00, 57.6MB/s]
validation-00000-of-00001.parquet: 100%  7.96M/7.96M [00:00<00:00, 42.7MB/s]
test-00000-of-00001.parquet: 100%  7.96M/7.96M [00:00<00:00, 28.0MB/s]
Generating train split: 100%  156915/156915 [00:00<00:00, 228029.73 examples/s]
Generating validation split: 100%  19614/19614 [00:00<00:00, 135985.73 examples/s]
Generating test split: 100%  19615/19615 [00:00<00:00, 171856.41 examples/s]
DatasetDict({
    train: Dataset({
        features: ['text'],
        num_rows: 156915
    })
    validation: Dataset({
        features: ['text'],
        num_rows: 19614
    })
    test: Dataset({
        features: ['text'],
        num_rows: 19615
    })
})
```

```
# getting token lengths from training dataset
train_length = []
train_token_length = []
for item in qa_dataset['train']:
  if item['text']:
    train_length.append(len(item['text']))
    tokens = tokenizer.encode(item['text'])
    train_token_length.append(len(tokens))
  else:
    print(item)
```

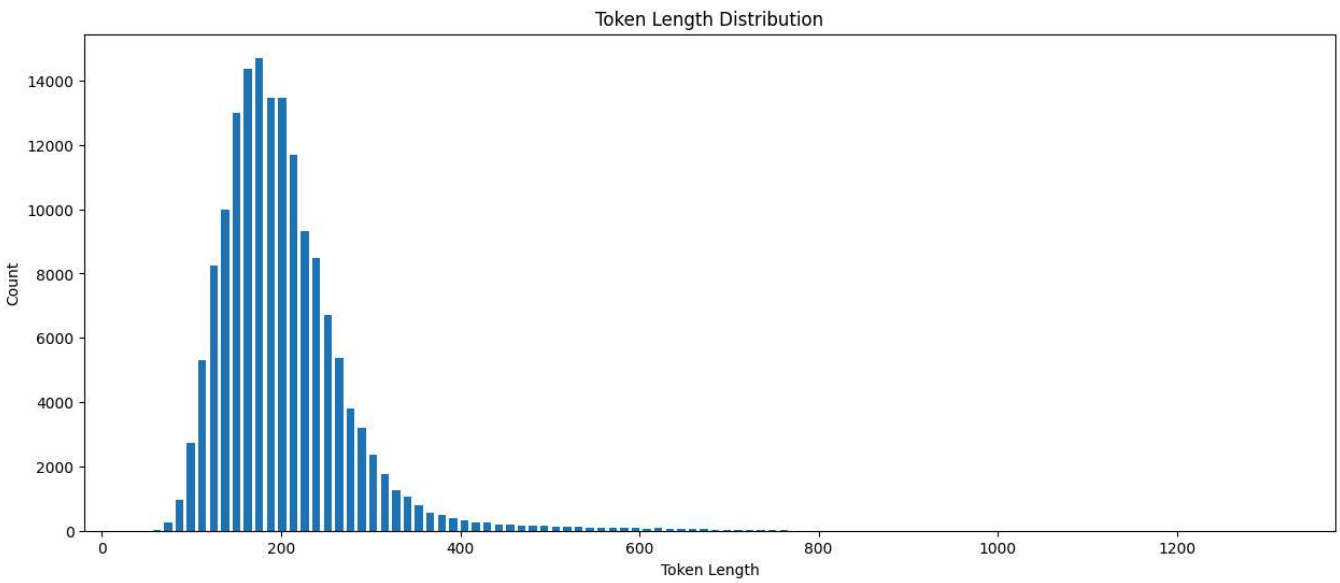
```
len(train_token_length)
```

```
156915
```

```
# print the min, max and average character lengths and token count lengths for training data
print(min(train_length), max(train_length), sum(train_length)/len(train_length))
print(min(train_token_length), max(train_token_length), sum(train_token_length)/len(train_token_length))
```

```
138 4677 786.0767421852595
42 1315 201.8653156167352
```

```
# plot the token lengths
plt.figure(figsize=(15, 6))
plt.hist(train_token_length, rwidth=0.7, bins=100)
plt.xlabel('Token Length')
plt.ylabel('Count')
plt.title('Token Length Distribution')
plt.show()
```




```
# avg is now approx 104, but will keep 230 tokens, therefore will create new datasets
# to filter out any rows above this token count
MAX_TOKENS = 250
DATASET_SIZE = 1000
def get_filtered_dataset(dataset, max_tokens, size):
  count = 0
  train_size = round(size * 0.9) # 90% for train
  valid_size = round(size * 0.1) # 10% for eval
  print(f"Train size: {train_size}, Eval size: {valid_size}")

  filtered_train_dataset = []
  for item in dataset['train']:
    if item['text']:
      tokens = tokenizer.encode(item['text'])
      if len(tokens) < max_tokens:
        filtered_train_dataset.append(item)
        count += 1
      if count >= train_size:
        break

  count = 0
  filtered_eval_dataset = []
  for item in dataset['validation']:
    if item['text']:
      tokens = tokenizer.encode(item['text'])
      if len(tokens) < max_tokens:
        filtered_eval_dataset.append(item)

        count += 1
      if count >= valid_size:
        break

  return filtered_train_dataset, filtered_eval_dataset
```

```
qa_train, qa_val = get_filtered_dataset(qa_dataset, MAX_TOKENS, DATASET_SIZE)
qa_train = Dataset.from_list(qa_train)
```

```
qa_val = Dataset.from_list(qa_val)
print(qa_train)
print(qa_val)
```

```
Train size: 900, Eval size: 100
Dataset({
    features: ['text'],
    num_rows: 900
})
Dataset({
    features: ['text'],
    num_rows: 100
})
```

```
qa_val[10]
```

```
{'text': '<s>[INST] Given this context: More than one half of lower respiratory cultures are negative for ventilator-associated pneumonia (VAP) and final reporting requires 72 hours to 96 hours. A previous retrospective study concluded that preliminary bronchoalveolar lavage (BAL) culture results (pBAL), reported at approximately 24 hours, accurately predicted final BAL culture results (fBAL). Our objective was to verify the predictive value of pBALs for fBALs, and evaluate the use of insignificant (1-99,999 cfu/mL) pBALs for rapid discontinuation of empirical antibiotics. Question: Does utility of preliminary bronchoalveolar lavage result in suspected ventilator-associated pneumonia? [/INST] Preliminary BALs were highly predictive for final results. Using insignificant pBALs appears to be a safe strategy for promptly discontinuing empirical antibiotics in trauma patients with suspected VAP. </s>'}
```

```
# set constants
MAX_SEQUENCE_LENGTH = 230 # calculated from avg token lenght in dataset

# Run name for saving the model in the hub
RUN_NAME =  f"{datetime.now():%Y-%m-%d_%H.%M.%S}"
PROJECT_RUN_NAME = f"{project_name}-{RUN_NAME}"
HUB_MODEL_NAME = f"{hf_username}/{PROJECT_RUN_NAME}"

# qlora hyper params
LORA_R = 16 # Reduce LoRA rank (lower = less memory) , initial = 32
LORA_ALPHA = 32 # Lower alpha , initial = 64
TARGET_MODULES = ["q_proj", "k_proj", "v_proj", "o_proj"]
LORA_DROPOUT = 0.05
QUANT_4_BIT = True

# training hyper params
EPOCHS = 3
BATCH_SIZE = 4  # Reduce batch size , initial = 4
GRADIENT_ACCUMULATION_STEPS = 8  # Simulate batch size 8, initial = 1
LEARNING_RATE = 2e-4
LR_SCHEDULER_TYPE = 'cosine'
WARMUP_RATIO = 0.03
OPTIMIZER = "paged_adamw_32bit"
STEPS = 50
SAVE_STEPS = 2000
LOG_TO_WANDB = True
```

```
response_template = " [/INST] "
collator = DataCollatorForCompletionOnlyLM(response_template, tokenizer=tokenizer)
```

```
lora_params = LoraConfig(
    r=LORA_R,
    lora_alpha=LORA_ALPHA,
    lora_dropout=LORA_DROPOUT,
    target_modules=TARGET_MODULES,
    bias="none",
    task_type="CAUSAL_LM"
)
```

```
training_params = SFTConfig(
    output_dir=PROJECT_RUN_NAME,
    num_train_epochs=EPOCHS,
    per_device_train_batch_size=BATCH_SIZE,
    per_device_eval_batch_size=1,
    #eval_strategy="no",
    gradient_accumulation_steps=GRADIENT_ACCUMULATION_STEPS,
    optim=OPTIMIZER,
    save_steps=SAVE_STEPS,
    save_total_limit=10,
    logging_steps=STEPS,
    learning_rate=LEARNING_RATE,
    weight_decay=0.001,
    fp16=False,
    bf16=True,
    max_grad_norm=0.3,
```

```
    max_steps=-1,
    warmup_ratio=WARMUP_RATIO,
    group_by_length=True,
    lr_scheduler_type=LR_SCHEDULER_TYPE,
    report_to="wandb" if True else None,
    run_name=RUN_NAME,
    max_seq_length=MAX_SEQUENCE_LENGTH,
    dataset_text_field="text",
    save_strategy="steps",
    hub_strategy="every_save",
    push_to_hub=True,
    hub_model_id=HUB_MODEL_NAME,
    hub_private_repo=True
    #neftune_noise_alpha=5 # using NEFTune as describe in SFT Trainer docs for increased conversational quality
)
```

```
!nvidia-smi
```

```
Sun May 11 04:10:46 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   68C    P0             32W /   70W |    6652MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
+-----------------------------------------------------------------------------------------+
```

```
# clear gpu memory
torch.cuda.empty_cache()
torch.cuda.reset_max_memory_allocated()
```

```
/usr/local/lib/python3.11/dist-packages/torch/cuda/memory.py:391: FutureWarning: torch.cuda.reset_max_memory_allocated now calls tor
  warnings.warn(
```

```
# Prepare the model for k-bit training (important for 4-bit)
model = prepare_model_for_kbit_training(model)
```

```
model = get_peft_model(model, lora_params)
```

```
trainer = SFTTrainer(
    model=model,
    train_dataset=qa_train,
    eval_dataset=qa_val,
    peft_config=lora_params,
    args=training_params
)

```

```
Converting train dataset to ChatML: 100%  900/900 [00:00<00:00, 12383.29 examples/s]
Adding EOS to train dataset: 100%  900/900 [00:00<00:00, 12465.28 examples/s]
Tokenizing train dataset: 100%  900/900 [00:00<00:00, 1677.64 examples/s]
Truncating train dataset: 100%  900/900 [00:00<00:00, 32253.95 examples/s]
Converting eval dataset to ChatML: 100%  100/100 [00:00<00:00, 4262.20 examples/s]
Adding EOS to eval dataset: 100%  100/100 [00:00<00:00, 4069.10 examples/s]
Tokenizing eval dataset: 100%  100/100 [00:00<00:00, 1147.91 examples/s]
Truncating eval dataset: 100%  100/100 [00:00<00:00, 5246.68 examples/s]
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names
```

Double-click (or enter) to edit

```
#torch.cuda.empty_cache()
trainer.train()
trainer.model.push_to_hub(PROJECT_RUN_NAME, private=True)
print(f"Model saved to HuggingFace as: {PROJECT_RUN_NAME}")
model.save_pretrained(PROJECT_RUN_NAME)

```

```
Changes to your `wandb` environment variables will be ignored because your `wandb` session has already started. For more information on how to modify your settings with `wandb.init()` arguments, please refer to the W&B docs.
Tracking run with wandb version 0.19.10
Run data is saved locally in /content/wandb/run-20250511_041757-hei1h23j
Syncing run 2025-05-11_04.05.52 to Weights & Biases (docs)
View project at https://wandb.ai/digitalblue-ai/ed_medical
View run at https://wandb.ai/digitalblue-ai/ed_medical/runs/hei1h23j
wandb: WARNING The get_url method is deprecated and will be removed in a future release. Please use `run.url` instead.
`use_cache=True` is incompatible with gradient checkpointing. Setting `use_cache=False`.
/usr/local/lib/python3.11/dist-packages/torch/_dynamo/eval_frame.py:745: UserWarning: torch.utils.checkpoint: the use_reentrant para
  return fn(*args, **kwargs)
[84/84 2:46:10, Epoch 2/3]
```

Step	Training Loss
50	1.693400

```
wandb: Adding directory to artifact (./ed_medical-2025-05-11_04.05.52/checkpoint-84)... Done. 0.6s
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names
README.md: 100%                                                  1.52k/1.52k [00:00<00:00, 93.3kB/s]
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 saved to HuggingFace as: ed medical-2025-05-11 04.05.52
```

```
results = trainer.evaluate()
print(results)

```

```
[100/100 02:50]
{'eval_loss': 1.4443223476409912, 'eval_runtime': 172.7306, 'eval_samples_per_second': 0.579, 'eval_steps_per_second': 0.579}
```

```
wandb.finish()
```

Run history:

eval/loss	_
eval/runtime	_
eval/samples_per_second	_
eval/steps_per_second	_
train/epoch	_▁▁
train/global_step	_▁▁
train/grad_norm	_
train/learning_rate	_
train/loss	_
train/mean_token_accuracy	_▁▁
train/num_tokens	_▁▁

Run summary:

eval/loss	1.44432
eval/runtime	172.7306
eval/samples_per_second	0.579
eval/steps_per_second	0.579
total_flos	1.8483999762579456e+16
train/epoch	2.92444
train/global_step	84
train/grad_norm	0.26786
train/learning_rate	8e-05
train/loss	1.6934
train/mean_token_accuracy	0.66989
train/num_tokens	480088
train_loss	1.57578
train_runtime	10115.7149
train_samples_per_second	0.267
train_steps_per_second	0.008

View run **2025-05-11_04.05.52** at: https://wandb.ai/digitalblue-ai/ed_medical/runs/hei1h23j
View project at: https://wandb.ai/digitalblue-ai/ed_medical
Synced 5 W&B file(s), 0 media file(s), 17 artifact file(s) and 0 other file(s)
Find logs at: ./wandb/run-20250511_041757-hei1h23j/logs

```
!zip -r ed_medical-2025-05-11_04.05.52.zip ed_medical-2025-05-11_04.05.52
```

```
  adding: ed_medical-2025-05-11_04.05.52/ (stored 0%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/ (stored 0%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/rng_state.pth (deflated 25%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/adapter_model.safetensors (deflated 7%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/training_args.bin (deflated 51%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/optimizer.pt (deflated 9%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/trainer_state.json (deflated 55%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/special_tokens_map.json (deflated 74%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/tokenizer.model (deflated 55%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/scheduler.pt (deflated 57%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/tokenizer.json (deflated 85%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/tokenizer_config.json (deflated 66%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/adapter_config.json (deflated 55%)
  adding: ed_medical-2025-05-11_04.05.52/checkpoint-84/README.md (deflated 66%)
  adding: ed_medical-2025-05-11_04.05.52/adapter_model.safetensors (deflated 7%)
  adding: ed_medical-2025-05-11_04.05.52/training_args.bin (deflated 51%)
  adding: ed_medical-2025-05-11_04.05.52/special_tokens_map.json (deflated 74%)
  adding: ed_medical-2025-05-11_04.05.52/tokenizer.model (deflated 55%)
  adding: ed_medical-2025-05-11_04.05.52/tokenizer.json (deflated 85%)
  adding: ed_medical-2025-05-11_04.05.52/tokenizer_config.json (deflated 66%)
  adding: ed_medical-2025-05-11_04.05.52/adapter_config.json (deflated 55%)
  adding: ed_medical-2025-05-11_04.05.52/README.md (deflated 46%)
```

```
!ls -la
```

```
total 242888
drwxr-xr-x 1 root root      4096 May 11 07:10 .
drwxr-xr-x 1 root root      4096 May 11 03:49 ..
drwxr-xr-x 4 root root      4096 May  8 13:38 .config
drwx------ 6 root root      4096 May 11 03:50 drive
drwxr-xr-x 3 root root      4096 May 11 07:06 ed_medical-2025-05-11_04.05.52
-rw-r--r-- 1 root root 248680821 May 11 07:10 ed_medical-2025-05-11_04.05.52.zip
drwxr-xr-x 1 root root      4096 May  8 13:38 sample_data
drwxr-xr-x 3 root root      4096 May 11 04:17 wandb
```

```
!cp ed_medical-2025-05-11_04.05.52.zip /content/drive/MyDrive/
```

```
# stop notebook and disconnect GPU after finishing above steps as this process can take several hours
# from google.colab import runtime
# runtime.unassign()
```

5. Evaluate the models - generate evaluation data

At this step the fine-tuned model created in the previous step can be re-loaded without need to re-run step 4. Generated responses will be collected, in addition to the base model to use as a benchmark.

The prompts to generate the responses are 50 samples from the **test** set, which was not used for fine-tuning.

The generated response for each model are then saved to HuggingFace hub for later analysis.

```
# define models saved to huggingface hub from previous step
MODEL_A = "digitalblue/ed_medical-2025-04-03_09.11.29" # trained on 200 rows
MODEL_B = "digitalblue/ed_medical-2025-04-05_10.59.54" # trained on 800 rows
MODEL_C = "digitalblue/ed_medical-2025-04-05_11.36.32" # trained on 1600 rows
MODEL_D = "digitalblue/ed_medical-2025-04-06_10.40.29" # trained on 1600 rows with NEFTune
MODEL_E = "digitalblue/ed_medical-2025-04-15_12.43.48" # trained on 8000 rows of pubmedqa only w/ NEFTune
MODEL_F = "digitalblue/ed_medical-2025-05-05_23.51.25" # trained on 9000 row, lower learing rate, 3 epochs
MODEL_G = "digitalblue/ed_medical-2025-05-08_01.23.37" # trained on 9000 rows, NEFTune false
MODEL_H = "digitalblue/ed_medical-2025-05-11_04.05.52" # trained on 900 row, with context, 3 epochs
```

```
# NOTE: Only using model E, F, G, H
```

```
# quantisation config to use less memory when loading model
quant_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.bfloat16,
    bnb_4bit_quant_type="nf4"
)
```

```
model_base = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=quant_config,
    device_map="auto")
```

```
Loading checkpoint shards: 100%  2/2 [01:28<00:00, 40.27s/it]
```

```
# initialise tokenizer and pipeline
tokenizer = AutoTokenizer.from_pretrained(model_name, token=hf_token)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
#qa_test = load_dataset("digitalblue/ed_medical", split="test[:50]") # first 50 from test set
qa_test = load_dataset("digitalblue/ed_medical-pubmedqa-artifical", split="test[:50]") # load only pubmedqa YES rows, test set
```

```
# split into question and answer list
qa_list = []
for item in qa_test:
  prompt = item['text']
  question = re.search(r'\[INST\] (.*) \[/INST\]', prompt).group(1)
  response = re.search(r'\[/INST\] (.*) \</s\>', prompt).group(1)
  qa_list.append([question, response])
```

```
qa_list[20][0] # question
```

```
'Do n-glycans of human amniotic fluid transferrin stimulate progesterone production in human first trimester trophoblast cells in v
```

```
prompt1 = "You are a helpful personalised medical assistant. In your response do not include any personal names and keep your answer
#prompt2 = "You are a helpful personalised medical assistant. Provide a friendly personal response to the patients medical question

def get_model_responses(model_x, qa_list):
  model_responses = []
  for item in qa_list:
    question = item[0]
    # print(item)
    # print(question)
    # print("-----")
    messages = [
```

```
    {"role": "system", "content": prompt1},
    {"role": "user", "content": question}
  ]
  inputs = tokenizer.apply_chat_template(messages, return_tensors="pt").to("cuda")
  outputs = model_x.generate(inputs, max_new_tokens=250, temperature=0.1)
  response = tokenizer.decode(outputs[0], skip_special_tokens=False)
  answer = response.split('[/INST] ')[1]  # get text after /INST token
  answer = answer.replace('</s>', '') # remove trailing special token if present
  model_responses.append(answer)
 return model_responses
```

```
print(model_base.lm_head.weight[0, :10])
print(model_g.lm_head.weight[0, :10])
```

```
tensor([-0.0036,  0.0027, -0.0074, -0.0100,  0.0063,  0.0049, -0.0065,  0.0012,
        -0.0009,  0.0107], device='cuda:0', dtype=torch.float16)
tensor([-0.0036,  0.0027, -0.0074, -0.0100,  0.0063,  0.0049, -0.0065,  0.0012,
        -0.0009,  0.0107], device='cuda:0', dtype=torch.float16)
```

```
model_base_responses_pubmedqa = get_model_responses(model_base, qa_list)
```

```
prompt_base_pubmedqa_dataset = Dataset.from_dict({"text": model_base_responses_pubmedqa})
```

```
prompt_base_pubmedqa_dataset.push_to_hub(f"{hf_username}/model_base_responses_pubmedqa", private=True)
```

```
Uploading the dataset shards: 100%  1/1 [00:01<00:00,  1.36s/it]
Creating parquet from Arrow format: 100%  1/1 [00:00<00:00, 110.81ba/s]
CommitInfo(commit_url='https://huggingface.co/datasets/digitalblue/model_base_responses_pubmedqa/commit/e7d5da1e7221601b91e115833b9
commit_message='Upload dataset', commit_description='', oid='e7d5da1e7221601b91e115833b91cd02cf1c0bd4', pr_url=None,
repo_url=RepoUrl('https://huggingface.co/datasets/digitalblue/model_base_responses_pubmedqa', endpoint='https://huggingface.co',
```

```
prompt_base_pubmedqa_dataset[1]
```

```
{'text': 'Dose interval between 2 and 4 hours is recommended to achieve the highest bioavailability. '}
```

```
model_e = PeftModel.from_pretrained(model_base, MODEL_E)
model_f = PeftModel.from_pretrained(model_base, MODEL_F)
```

```
model_e_responses = get_model_responses(model_e, qa_list)
```

```
model_f_responses = get_model_responses(model_f, qa_list)
```

```
prompt_e_dataset = Dataset.from_dict({"text": model_e_responses})
prompt_f_dataset = Dataset.from_dict({"text": model_f_responses})
```

```
prompt_e_dataset.push_to_hub(f"{hf_username}/model_e_responses", private=True)
prompt_f_dataset.push_to_hub(f"{hf_username}/model_f_responses", private=True)
```

```
Uploading the dataset shards: 100%  1/1 [00:01<00:00,  1.37s/it]
Creating parquet from Arrow format: 100%  1/1 [00:00<00:00, 112.82ba/s]
Uploading the dataset shards: 100%  1/1 [00:01<00:00,  1.46s/it]
Creating parquet from Arrow format: 100%  1/1 [00:00<00:00, 117.08ba/s]
CommitInfo(commit_url='https://huggingface.co/datasets/digitalblue/model_f_responses/commit/8ac2484b0737fdde8a0f5f87d6471242afca9fd
commit_message='Upload dataset', commit_description='', oid='8ac2484b0737fdde8a0f5f87d6471242afca9fd6', pr_url=None,
repo_url=RepoUrl('https://huggingface.co/datasets/digitalblue/model_f_responses', endpoint='https://huggingface.co',
```

```
model_g = PeftModel.from_pretrained(model_base, MODEL_G)
```

```
adapter_config.json: 100%  850/850 [00:00<00:00, 38.3kB/s]
adapter_model.safetensors: 100%  793M/793M [00:34<00:00, 21.9MB/s]
```

```
model_g_responses = get_model_responses(model_g, qa_list)
prompt_g_dataset = Dataset.from_dict({"text": model_g_responses})
prompt_g_dataset.push_to_hub(f"{hf_username}/model_g_responses", private=True)
```

```
Uploading the dataset shards: 100%  1/1 [00:01<00:00,  1.42s/it]
Creating parquet from Arrow format: 100%  1/1 [00:00<00:00, 78.17ba/s]
CommitInfo(commit_url='https://huggingface.co/datasets/digitalblue/model_g_responses/commit/18661dc8f0a4129ddbce079efe62700c8f909c5d',
commit_message='Upload dataset', commit_description='', oid='18661dc8f0a4129ddbce079efe62700c8f909c5d', pr_url=None,
repo_url=RepoUrl('https://huggingface.co/datasets/digitalblue/model_g_responses', endpoint='https://huggingface.co',
```

```
model_h = PeftModel.from_pretrained(model_base, MODEL_H)
model_h_responses = get_model_responses(model_h, qa_list)
prompt_h_dataset = Dataset.from_dict({"text": model_h_responses})
prompt_h_dataset.push_to_hub(f"{hf_username}/model_h_responses", private=True)
```

```
Uploading the dataset shards: 100%  1/1 [00:01<00:00,  1.43s/it]
Creating parquet from Arrow format: 100%  1/1 [00:00<00:00, 39.14ba/s]
CommitInfo(commit_url='https://huggingface.co/datasets/digitalblue/model_h_responses/commit/06fce8700b9d76ff82f9434c13399375f50ce198',
commit_message='Upload dataset', commit_description='', oid='06fce8700b9d76ff82f9434c13399375f50ce198', pr_url=None,
repo_url=RepoUrl('https://huggingface.co/datasets/digitalblue/model_h_responses', endpoint='https://huggingface.co',
```

```
# stop notebook and disconnect GPU after finishing above steps as this process can take several hours
from google.colab import runtime
runtime.unassign()
```

6. Evaluate fine tuned models - get metrics from generated responses

In this final step, the generated response can be reloaded from HuggingFace to evaluate and calculate metrics against the expected responses in **test** dataset.

```
qa_test = load_dataset("digitalblue/ed_medical-pubmedqa-artifical", split="test[:50]") # load only pubmedqa YES rows, test set
```

```
qa_list = []
for item in qa_test:
  prompt = item['text']
  question = re.search(r'\[INST\] (.*) \[/INST\]', prompt).group(1)
  response = re.search(r'\[/INST\] (.*) \</s\>', prompt).group(1)
  qa_list.append([question, response])
```

```
qa_list[20][1] # index 1 is dataset real response
```

```
'The results suggest that amnion-transferrin and especially its N-glycans modulate the endocrine function of trophoblasts in cultur
```

```
qa_references = [item[1] for item in qa_list] # get list of reference responses
```

```
# load saved response data to evaluate
model_base_pubmedqa_dataset = load_dataset("digitalblue/model_base_responses_pubmedqa")
model_e_dataset = load_dataset("digitalblue/model_e_responses")
model_f_dataset = load_dataset("digitalblue/model_f_responses")
model_g_dataset = load_dataset("digitalblue/model_g_responses")
model_h_dataset = load_dataset("digitalblue/model_h_responses")
```

```
README.md: 100%                                  267/267 [00:00<00:00, 17.3kB/s]
train-00000-of-00001.parquet: 100%               11.1k/11.1k [00:00<00:00, 1.12MB/s]
Generating train split: 100%                     100/100 [00:00<00:00, 1707.56 examples/s]
README.md: 100%                                  267/267 [00:00<00:00, 27.1kB/s]
train-00000-of-00001.parquet: 100%               11.1k/11.1k [00:00<00:00, 1.11MB/s]
Generating train split: 100%                     100/100 [00:00<00:00, 5452.25 examples/s]
README.md: 100%                                  267/267 [00:00<00:00, 28.1kB/s]
train-00000-of-00001.parquet: 100%               11.3k/11.3k [00:00<00:00, 1.23MB/s]
Generating train split: 100%                     100/100 [00:00<00:00, 4327.95 examples/s]
README.md: 100%                                  266/266 [00:00<00:00, 27.9kB/s]
train-00000-of-00001.parquet: 100%               25.4k/25.4k [00:00<00:00, 2.31MB/s]
Generating train split: 100%                     50/50 [00:00<00:00, 2815.42 examples/s]
README.md: 100%                                  263/263 [00:00<00:00, 18.5kB/s]
train-00000-of-00001.parquet: 100%               5.44k/5.44k [00:00<00:00, 611kB/s]
Generating train split: 100%                     50/50 [00:00<00:00, 2617.35 examples/s]
```

```
# convert to list
base_pubmedqa_response_list = model_base_pubmedqa_dataset['train']['text'][:50]
e_response_list = model_e_dataset['train']['text'][:50]
f_response_list = model_f_dataset['train']['text'][:50]
g_response_list = model_g_dataset['train']['text']
h_response_list = model_h_dataset['train']['text']
```

```
def calc_rouge_metric(references, predictions):
  rouge = evaluate.load('rouge')
  results = rouge.compute(
    predictions=predictions,
    references=references
  )
  return results

def calc_bleu_metric(references, predictions):
  bleu = evaluate.load('bleu')
  results = bleu.compute(
    predictions=predictions,
    references=references
  )
  return results

def calc_meteor_metric(references, predictions):
  meteor = evaluate.load('meteor')
  results = meteor.compute(
    predictions=predictions,
    references=[[ref] for ref in references]
  )
  return results

def calc_bert_score(references, predictions):
  bert_scorer = BERTScorer(model_type='bert-base-uncased')
  P = []
  R = []
  F1 = []

  for i in range(len(predictions)):
    p_i, r_i, f1_i = bert_scorer.score([predictions[i]], [references[i]])
    P.append(p_i)
    R.append(r_i)
    F1.append(f1_i)
  results = {
    'P': np.mean(P),
    'R': np.mean(R),
    'F1': np.mean(F1)
  }
  return results
```

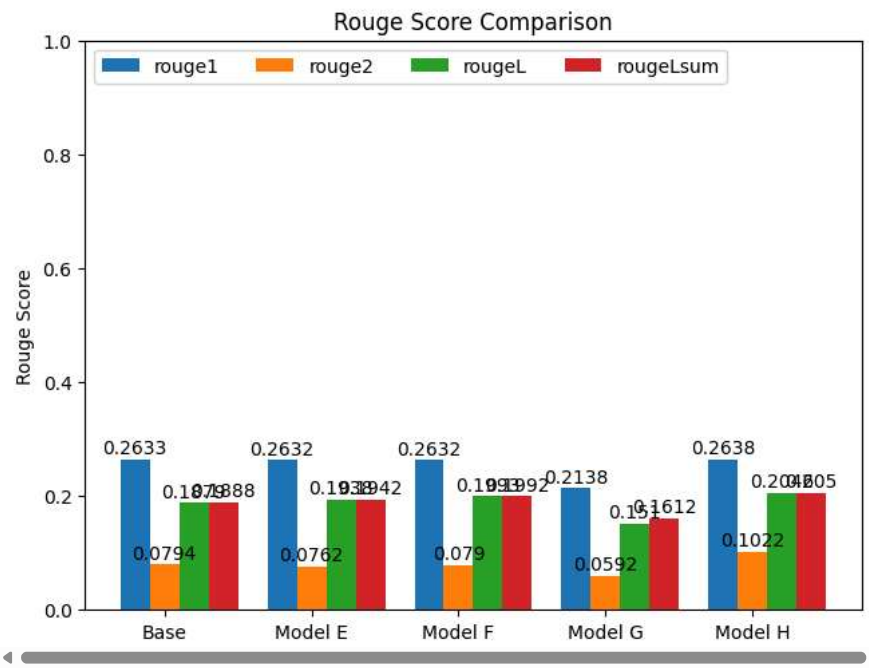
```
rouge_base_pubmedqa = calc_rouge_metric(qa_references, base_pubmedqa_response_list)
rouge_e = calc_rouge_metric(qa_references, e_response_list)
rouge_f = calc_rouge_metric(qa_references, f_response_list)
rouge_g = calc_rouge_metric(qa_references, g_response_list)
rouge_h = calc_rouge_metric(qa_references, h_response_list)
```

```
Downloading builder script: 100%  6.27k/6.27k [00:00<00:00, 640kB/s]
```

```
print(rouge_base_pubmedqa)
print(rouge_e)
print(rouge_f)
print(rouge_g)
print(rouge_h)
```

```
{'rouge1': np.float64(0.26328473964307153), 'rouge2': np.float64(0.07937912538217656), 'rougeL': np.float64(0.18786841702419282), 'r
{'rouge1': np.float64(0.263218379753081), 'rouge2': np.float64(0.07624340373229432), 'rougeL': np.float64(0.19380980070682116), 'rou
{'rouge1': np.float64(0.263197307016511), 'rouge2': np.float64(0.07898295649206828), 'rougeL': np.float64(0.19925342907701712), 'rou
{'rouge1': np.float64(0.2138327372000946), 'rouge2': np.float64(0.059210807794686446), 'rougeL': np.float64(0.15098541311199265), 'r
{'rouge1': np.float64(0.2637618500720388), 'rouge2': np.float64(0.10223797199683642), 'rougeL': np.float64(0.2046431306067485), 'rou
```

```
# plot rouge scores
rouge_dict = {
    'rouge1': (rouge_base_pubmedqa['rouge1'], rouge_e['rouge1'], rouge_f['rouge1'], rouge_g['rouge1'], rouge_h['rouge1']),
    'rouge2': (rouge_base_pubmedqa['rouge2'], rouge_e['rouge2'], rouge_f['rouge2'], rouge_g['rouge2'], rouge_h['rouge2']),
    'rougeL': (rouge_base_pubmedqa['rougeL'], rouge_e['rougeL'], rouge_f['rougeL'], rouge_g['rougeL'], rouge_h['rougeL']),
    'rougeLsum': (rouge_base_pubmedqa['rougeLsum'], rouge_e['rougeLsum'], rouge_f['rougeLsum'], rouge_g['rougeLsum'], rouge_h['rouge
}
model_labels = ('Base', 'Model E', 'Model F', 'Model G', 'Model H')

x = np.arange(len(model_labels))
width = 0.2
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in rouge_dict.items():
    offset = width * multiplier
    msr = np.round(measurement, 4) # round to 4 decimal places for plot
    rects = ax.bar(x + offset, msr, width, label=attribute)
    ax.bar_label(rects, padding=1)
    multiplier += 1

ax.set_ylabel('Rouge Score')
ax.set_title('Rouge Score Comparison')
ax.set_xticks(x + width, model_labels)
ax.legend(loc='upper left', ncols=4)
ax.set_ylim(0, 1)

plt.show()
```




```
bleu_base_pubmedqa = calc_bleu_metric(qa_references, base_pubmedqa_response_list)
bleu_e = calc_bleu_metric(qa_references, e_response_list)
```

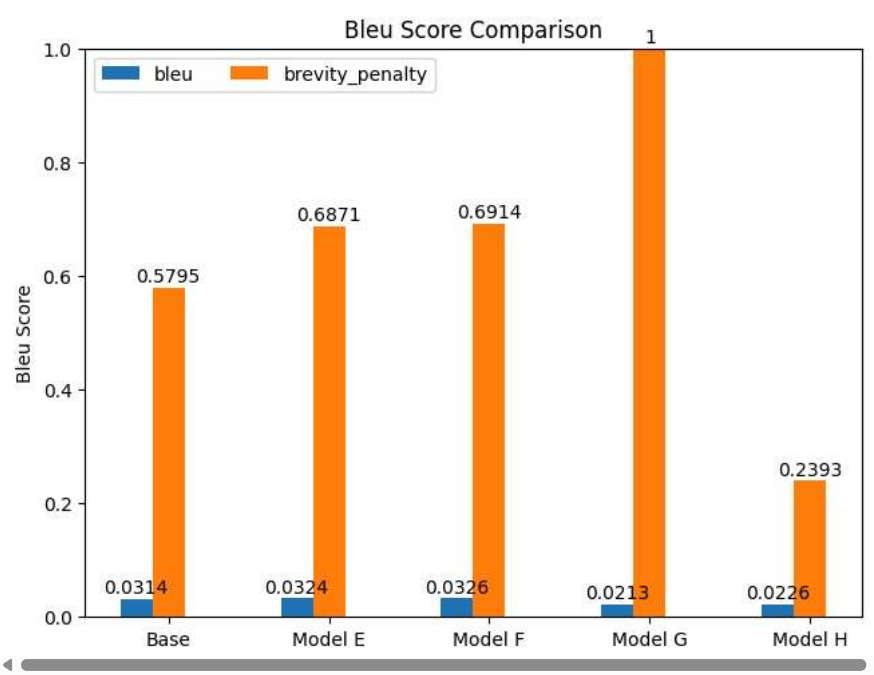
```
bleu_f = calc_bleu_metric(qa_references, f_response_list)
bleu_g = calc_bleu_metric(qa_references, g_response_list)
bleu_h = calc_bleu_metric(qa_references, h_response_list)
```

```
Downloading builder script: 100%  5.94k/5.94k [00:00<00:00, 106kB/s]
Downloading extra modules:  4.07k/? [00:00<00:00, 99.1kB/s]
Downloading extra modules: 100%  3.34k/3.34k [00:00<00:00, 356kB/s]
```

```
print(bleu_base_pubmedqa)
print(bleu_e)
print(bleu_f)
print(bleu_g)
print(bleu_h)
```

```
{'bleu': 0.03138988119399103, 'precisions': [0.3380281690140845, 0.07929176289453425, 0.029623698959167333, 0.010842368640533779],
{'bleu': 0.032379811967089916, 'precisions': [0.3113456464379947, 0.07094133697135062, 0.025423728813559324, 0.008784773060029283],
{'bleu': 0.0326166306491595, 'precisions': [0.3112278397898884, 0.07196198234894773, 0.025298664792691498, 0.00873998543335761], 'br
{'bleu': 0.021269429662351482, 'precisions': [0.1367226770070372, 0.032402719513959205, 0.01049103890426927, 0.004403346543372964],
{'bleu': 0.022637121921975846, 'precisions': [0.40326340326340326, 0.12128712871287128, 0.052770448548812667, 0.031029619181946404],
```

```
# plot bleu scores
bleu_dict = {
    'bleu': (bleu_base_pubmedqa['bleu'], bleu_e['bleu'], bleu_f['bleu'], bleu_g['bleu'], bleu_h['bleu']),
    'brevity_penalty': (bleu_base_pubmedqa['brevity_penalty'], bleu_e['brevity_penalty'], bleu_f['brevity_penalty'], bleu_g['brevity
}

model_labels = ('Base', 'Model E', 'Model F', 'Model G', 'Model H')

x = np.arange(len(model_labels))
width = 0.2
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in bleu_dict.items():
    offset = width * multiplier
    msr = np.round(measurement, 4) # round to 4 decimal places for plot
    rects = ax.bar(x + offset, msr, width, label=attribute)
    ax.bar_label(rects, padding=1)
    multiplier += 1

ax.set_ylabel('Bleu Score')
ax.set_title('Bleu Score Comparison')
ax.set_xticks(x + width, model_labels)
ax.legend(loc='upper left', ncols=4)
ax.set_ylim(0, 1)

plt.show()
```




```
meteor_base_pubmedqa = calc_meteor_metric(qa_references, base_pubmedqa_response_list)
meteor_e = calc_meteor_metric(qa_references, e_response_list)
```

```
meteor_f = calc_meteor_metric(qa_references, f_response_list)
meteor_g = calc_meteor_metric(qa_references, g_response_list)
meteor_h = calc_meteor_metric(qa_references, h_response_list)
```

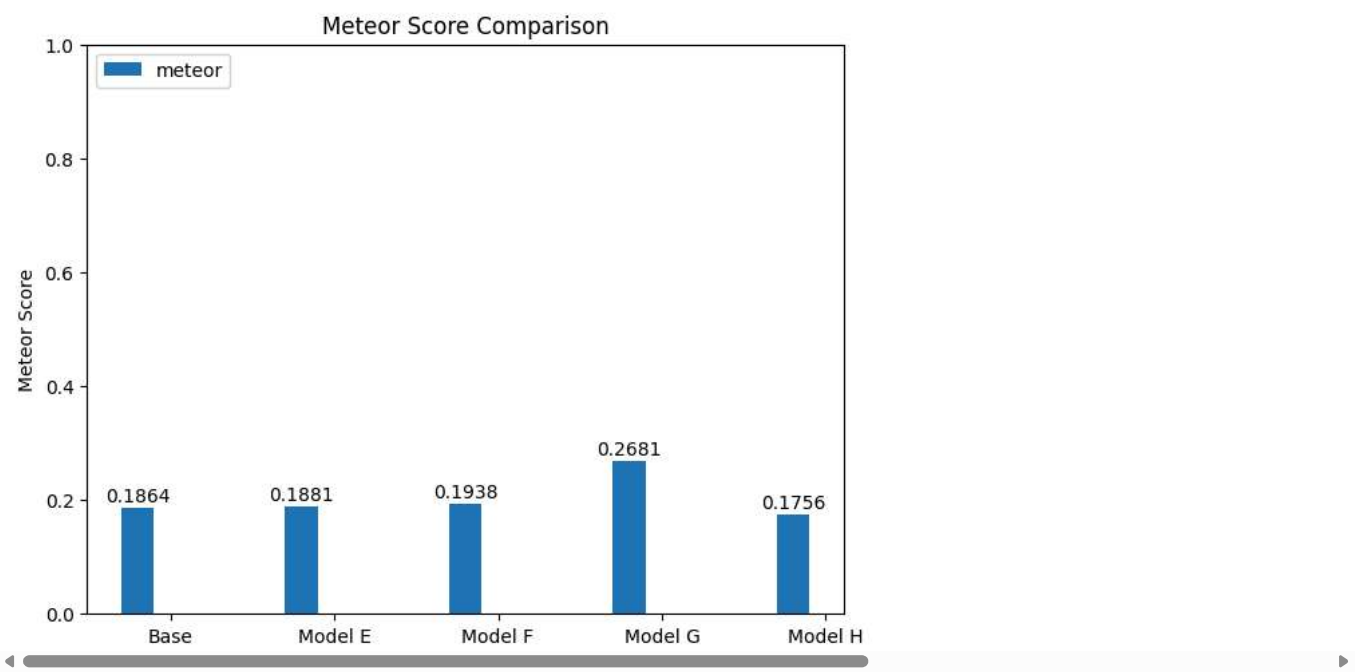
Downloading builder script: 100% 7.02k/7.02k [00:00<00:00, 481kB/s]

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
print(meteor_base_pubmedqa)
print(meteor_e)
print(meteor_f)
print(meteor_g)
print(meteor_h)
```

```
{'meteor': np.float64(0.18639960354363067)}
{'meteor': np.float64(0.1881447693596208)}
{'meteor': np.float64(0.19384352470409308)}
{'meteor': np.float64(0.26814099277140213)}
{'meteor': np.float64(0.17556055146287494)}
```

```
# plot meteor scores
meteor_dict = {
    'meteor': (meteor_base_pubmedqa['meteor'], meteor_e['meteor'], meteor_f['meteor'], meteor_g['meteor'], meteor_h['meteor'])
}
model_labels = ('Base', 'Model E', 'Model F', 'Model G', 'Model H')

x = np.arange(len(model_labels))
width = 0.2
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in meteor_dict.items():
    offset = width * multiplier
    msr = np.round(measurement, 4) # round to 4 decimal places for plot
    rects = ax.bar(x + offset, msr, width, label=attribute)
    ax.bar_label(rects, padding=1)
    multiplier += 1

ax.set_ylabel('Meteor Score')
ax.set_title('Meteor Score Comparison')
ax.set_xticks(x + width, model_labels)
ax.legend(loc='upper left', ncols=4)
ax.set_ylim(0, 1)

plt.show()
```




```
bert_score_base_pubmedqa = calc_bert_score(qa_references, base_pubmedqa_response_list)
bert_score_e = calc_bert_score(qa_references, e_response_list)
bert_score_f = calc_bert_score(qa_references, f_response_list)
bert_score_g = calc_bert_score(qa_references, g_response_list)
bert_score_h = calc_bert_score(qa_references, h_response_list)
```

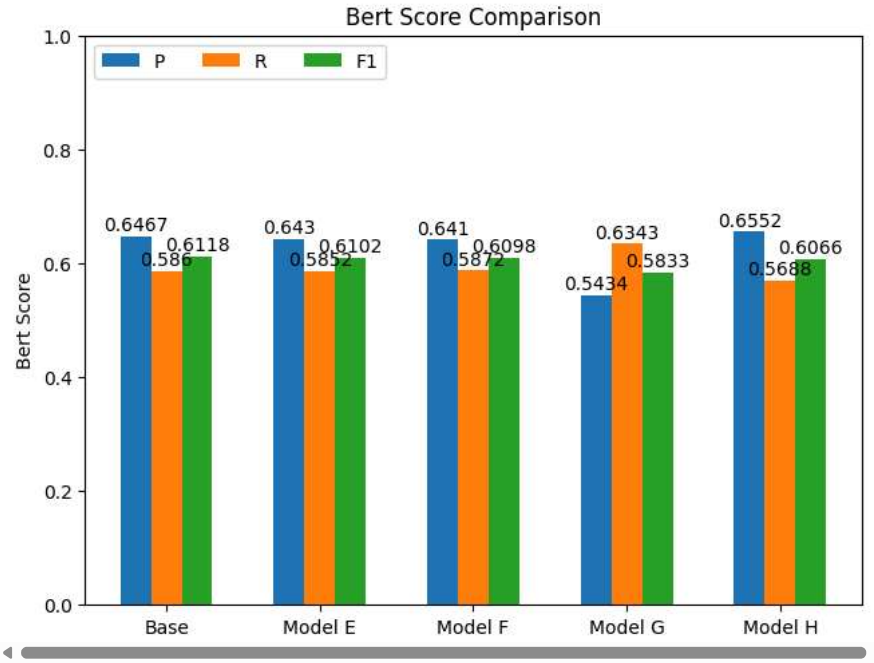
```
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 1.14kB/s]
config.json: 100%  570/570 [00:00<00:00, 16.0kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 19.6MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 2.15MB/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better p
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back
model.safetensors: 100%  440M/440M [00:01<00:00, 273MB/s]
```

```
print(bert_score_base_pubmedqa)
print(bert_score_e)
print(bert_score_f)
print(bert_score_g)
print(bert_score_h)
```

```
{'P': np.float32(0.6467173), 'R': np.float32(0.58598), 'F1': np.float32(0.6117929)}
{'P': np.float32(0.64296514), 'R': np.float32(0.585245), 'F1': np.float32(0.61016136)}
{'P': np.float32(0.64098334), 'R': np.float32(0.58719736), 'F1': np.float32(0.6097659)}
{'P': np.float32(0.5433609), 'R': np.float32(0.6343017), 'F1': np.float32(0.58330375)}
{'P': np.float32(0.6551912), 'R': np.float32(0.56881934), 'F1': np.float32(0.60660475)}
```

```
# plot bert scores
bert_score_dict = {
    'P': (bert_score_base_pubmedqa['P'], bert_score_e['P'], bert_score_f['P'], bert_score_g['P'], bert_score_h['P']),
    'R': (bert_score_base_pubmedqa['R'], bert_score_e['R'], bert_score_f['R'], bert_score_g['R'], bert_score_h['R']),
    'F1': (bert_score_base_pubmedqa['F1'], bert_score_e['F1'], bert_score_f['F1'], bert_score_g['F1'], bert_score_h['F1'])
}
model_labels = ('Base', 'Model E', 'Model F', 'Model G', 'Model H')

x = np.arange(len(model_labels))
width = 0.2
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in bert_score_dict.items():
    offset = width * multiplier
    msr = np.round(measurement, 4) # round to 4 decimal places for plot
    rects = ax.bar(x + offset, msr, width, label=attribute)
    ax.bar_label(rects, padding=1)
    multiplier += 1

ax.set_ylabel('Bert Score')
ax.set_title('Bert Score Comparison')
```

```
ax.set_xticks(x + width, model_labels)
ax.legend(loc='upper left', ncols=4)
ax.set_ylim(0, 1)

plt.show()
```




```
# human evaluation
def print_response(QA_NUM):
  for i in range(QA_NUM):
    print('----------------------------------------')
    print('\nTest question: ' + qa_list[i][0])
    print('\nTest answer: ' + qa_list[i][1])
    print('\nBase: ' + base_pubmedqa_response_list[i])
    print('\nModel E: ' + e_response_list[i])
    print('\nModel F: ' + f_response_list[i])
    print('\nModel G: ' + g_response_list[i])
    print('\nModel H: ' + h_response_list[i])

print_response(5)
```

```
----------------------------------------

Test question: Does simvastatin attenuate hepatic sensitization to lipopolysaccharide after partial hepatectomy?

Test answer: These novels findings demonstrate that simvastatin protects the remnant liver against endotoxemic injury following ma

Base: The results suggest that simvastatin attenuates the increase in the expression of CD14 and TNF-alpha after partial hepatecto

Model E: The results suggest that simvastatin attenuates the increase in mRNA expression of TNF-alpha and IL-1beta in the liver af

Model F: The results suggest that simvastatin attenuates the increase in mRNA expression of TNF-alpha and IL-1beta in the liver af

Model G:  Yes, simvastatin has been shown to attenuate hepatic sensitization to lipopolysaccharide (LPS) after partial hepatectomy

Hepatic sensitization to LPS is a common complication after partial hepatectomy, leading to increased inflammation and impaired li

Studies have shown that simvastatin can reduce the expression of toll-like receptor 4 (TLR4) and myeloid differentiation factor 88

Overall, the available evidence suggests that simvastatin may be

Model H: Simvastatin attenuates hepatic sensitization to lipopolysaccharide after partial hepatectomy.
----------------------------------------

Test question: Does mizoribine require individual dosing due to variation of bioavailability?

Test answer: Individual dosing is required to optimize C(max) in childhood-onset glomerular disease patients. The safe dosage of M

Base: Dose interval between 2 and 4 hours is recommended to achieve the highest bioavailability.

Model E: Dose interval between 2 and 4 hours is recommended to achieve the highest bioavailability.

Model F: Dear patient,

Model G:  Yes, mizoribine can exhibit significant inter-individual variability in bioavailability, which may require individualize

Model H: Mizoribine is a drug that requires individual dosing due to variation of bioavailability.
----------------------------------------
```

```
Test question: Does cold reperfusion before rewarming reduce neurological events after deep hypothermic circulatory arrest?

Test answer: sDHCA remains a safe and easy tool for cerebral protection when DHCA duration is expected to be less than 30 min. Whe

Base: This study suggests that cold reperfusion followed by rewarming may reduce the incidence of neurological events after DHCA.

Model E: This study suggests that cold reperfusion followed by rewarming may reduce neurological events after DHCA.

Model F: Cold reperfusion is associated with a significant reduction in neurological events after deep hypothermic circulatory arr

Model G:  Cold reperfusion before rewarming has been shown to reduce the incidence of neurological events after deep hypothermic c

Cold reperfusion, which involves rewarming the body at a slow rate, has been found to reduce the risk of neurological events such

However, it is important to note that the evidence for the benefit of cold reperfusion is not yet definitive, and more research is

Model H: The results of this study suggest that cold reperfusion before rewarming does not reduce the incidence of neurological ev
----------------------------------------
```

7. Evaluation Summary

Evaluation metrics do not show improvement over base model. The Llama 2 model does not disclose its training dataset, however as PubMedQA is public data, there exists the possiblity that it may have already been trained on this dataset.

Alternatively, the training hyperparameters used in training and/or the volume of data used may have been insufficinet for the model to successfully learn the domain.

Next steps:

- Continue to experiment with further hyperparameter tuning
- Increase volume of training data
- Adjust system prompt, and model parameters on inference
- Explore different dataset

---

8. Model packaging

This sections explored how to bundle the fine-tuned model with adapters and including tokenizer, into an LLM that can be downloaded and used standalone. The intention was to investigate how it could be deployed to a cloud provider, and used on edge device or local PC via Ollama.

```
# clear gpu memory
torch.cuda.empty_cache()
torch.cuda.reset_max_memory_allocated()
```

```
/usr/local/lib/python3.11/dist-packages/torch/cuda/memory.py:391: FutureWarning: torch.cuda.reset_max_memory_allocated now calls tor
  warnings.warn(
```

```
# quantisation config to use less memory when loading model
quant_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.bfloat16,
    bnb_4bit_quant_type="nf4"
)
```

```
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    #quantization_config=quant_config,
    device_map="auto"
    )
```

```
config.json: 100%   614/614 [00:00<00:00, 45.7kB/s]
model.safetensors.index.json: 100%   26.8k/26.8k [00:00<00:00, 2.64MB/s]
Fetching 2 files: 100%   2/2 [01:16<00:00, 76.74s/it]
model-00001-of-00002.safetensors: 100%   9.98G/9.98G [01:16<00:00, 87.7MB/s]
model-00002-of-00002.safetensors: 100%   3.50G/3.50G [00:38<00:00, 141MB/s]
Loading checkpoint shards: 100%   2/2 [00:58<00:00, 26.71s/it]
generation_config.json: 100%   188/188 [00:00<00:00, 9.34kB/s]
WARNING:accelerate.big_modeling:Some parameters are on the meta device because they were offloaded to the cpu and disk.
```

```
# initialise tokenizer and pipeline
tokenizer = AutoTokenizer.from_pretrained(model_name, token=hf_token)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
tokenizer_config.json: 100%   1.62k/1.62k [00:00<00:00, 84.0kB/s]
tokenizer.model: 100%   500k/500k [00:00<00:00, 25.5MB/s]
tokenizer.json: 100%   1.84M/1.84M [00:00<00:00, 4.13MB/s]
special_tokens_map.json: 100%   414/414 [00:00<00:00, 45.4kB/s]
```

```
MODEL_A = "digitalblue/ed_medical-2025-04-03_09.11.29" # trained on 200 rows
MODEL_B = "digitalblue/ed_medical-2025-04-05_10.59.54" # trained on 800 rows
MODEL_C = "digitalblue/ed_medical-2025-04-05_11.36.32" # trained on 1600 rows
MODEL_D = "digitalblue/ed_medical-2025-04-06_10.40.29" # trained on 1600 rows with NEFTune
MODEL_E = "digitalblue/ed_medical-2025-04-15_12.43.48" # trained on 8000 rows of pubmedqa only w/ NEFTune
MODEL_F = "digitalblue/ed_medical-2025-05-05_23.51.25" # trained on 9000 row, lower learing rate, 3 epochs
```

```
model_e = PeftModel.from_pretrained(model, MODEL_E)
```

```
adapter_config.json: 100%   754/754 [00:00<00:00, 56.4kB/s]
adapter_model.safetensors: 100%   134M/134M [00:00<00:00, 247MB/s]
```

```
print(model.lm_head.weight[0, :10])
print(model_e.lm_head.weight[0, :10])
print(model_f.lm_head.weight[0, :10])
```

```
tensor([-0.0115,  0.0007, -0.0078, -0.0107,  0.0119,  0.0019,  0.0006, -0.0050,
        -0.0156,  0.0156], device='cuda:0', dtype=torch.float16,
       grad_fn=<SliceBackward0>)
tensor([-0.0115,  0.0007, -0.0078, -0.0107,  0.0119,  0.0019,  0.0006, -0.0050,
        -0.0156,  0.0156], device='cuda:0', dtype=torch.float16,
       grad_fn=<SliceBackward0>)
tensor([-0.0115,  0.0007, -0.0078, -0.0107,  0.0119,  0.0019,  0.0006, -0.0050,
        -0.0156,  0.0156], device='cuda:0', dtype=torch.float16,
       grad_fn=<SliceBackward0>)
```

```
model_e.print_trainable_parameters()
model_f.print_trainable_parameters()
```

```
trainable params: 262,144,000 || all params: 7,008,948,224 || trainable%: 3.7401
trainable params: 262,144,000 || all params: 7,008,948,224 || trainable%: 3.7401
```

```
messages = [
    {"role": "system", "content": "You are a helpful personalised medical assistant. In your response do not include any personal na
    {"role": "user", "content": "Does methylphenidate improve the quality of life of children and adolescents with ADHD and difficul
]
```

```
og_output = model_e.generate(tokenizer.apply_chat_template(messages, return_tensors="pt").to("cuda"), max_new_tokens=250)
```

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
The attention mask is not set and cannot be inferred from input because pad token is same as eos token. As a consequence, you may ob
```

```
tokenizer.decode(og_output[0], skip_special_tokens=False)
```

```
'<s> [INST] <<SYS>>\nYou are a helpful personalised medical assistant. In your response do not include any personal names and keep
your answer professional and concise.\n<</SYS>>\n\nDoes methylphenidate improve the quality of life of children and adolescents wit
```

```
base_output = model.generate(tokenizer.apply_chat_template(messages, return_tensors="pt").to("cuda"), max_new_tokens=250)
tokenizer.decode(base_output[0], skip_special_tokens=False)
```

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
'<s> [INST] <<SYS>>\nYou are a helpful personalised medical assistant. In your response do not include any personal names and keep
your answer professional and concise.\n<</SYS>>\n\nDoes methylphenidate improve the quality of life of children and adolescents wit
```

```
merge_model_e = model_e.merge_and_unload()
```

```
/usr/local/lib/python3.11/dist-packages/peft/tuners/lora/bnb.py:351: UserWarning: Merge lora module to 4-bit linear may get differer
  warnings.warn(
```

```
merge_model_f = model_f.merge_and_unload()
```

```
merge_model_f
```

```
LlamaForCausalLM(
  (model): LlamaModel(
```