First Full LoRA Trial with Transformer

Starting with going through what I've done as well as finishing the task of getting my LoRA-fine-tuned model from Hugging Face and running inference on it (i.e. testing it using the test set). See the first timestamp below for the new timing. By the way, I've shut down and rebooted the compy here in the corner with the three screens).

peft (for LoRA) and FLAN-T5-small for the LLM

I'm following what seems to be a great tutorial from Mehul Gupta,

> https://medium.com/data-science-in-your-pocket/lora-for-fine-tuning-llms-explained-with-codes-and-example-62a7ac5a3578
>
> https://web.archive.org/web/20240522140323/https://medium.com/data-science-in-your-pocket/lora-for-fine-tuning-llms-explained-with-codes-and-example-62a7ac5a3578

I'm doing this to prepare creating a LoRA for RWKV (~~@todo~~ @DONE put links in here) so as to fine-tune it for Pat's OLECT-LM stuff.

In [13]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y%m%dT%H%M%S%Z00\") -replace '[.][0-9]*_',
```

Output was:

```
1717405690_20240603T090810-0600
```

Installation

My `environment.yml` file will have its contents listed below. It should have everything needed for an install anywhere. The directory should have a `full_environment.yml`, which includes everything for the environment on Windows.

You can change `do_want_to_read_realtime` to `True` if you really want to see the file as it is now. One case of this would be that you think `environment.yml` has been changed since this notebook was written. The file contents as of the time of my writing this notebook should be in a markdown cell beneath the code.

In [14]:
```
do_want_to_read_realtime = False

if do_want_to_read_realtime:
    with open("environment.yml", 'r', encoding='utf-8') as fh:
```

```
        while True:
            line = fh.readline()
            if not line:
                break
            ##endof:  if not line
            print(line.replace("\n", ""))
        ##endof:  while True
    ##endof:  with open ... fh
##endof:  if do_want_to_read_realtime
```

```
# @file: environment.yml
# @since 2024-06-03
## 1717411989_2024-06-03T105309-0600
## IMPORTANT NOTES
##
##  A couple of installations were made from git repos.
##     >pip install git+https://github.com/huggingface/peft.git
##     >pip install git+https://github.com/nexplorer-3e/qwqfetch
##
##  The commit info will be important for reproducibility.
##
##-----
##  qwqfetch   for system info
##
##   Resolved https://github.com/nexplorer-3e/qwqfetch \
##       to commit f72d222e2fff5ffea9f4e4b3a203e4c4d9e8cf00
##   Successfully installed qwqfetch-0.0.0
##
#
##-----
##  peft: I installed PEFT among other things, but I'm picking out
##+       stuff relevant to peft. PEFT has LoRA in it.
##
##   Resolved https://github.com/huggingface/peft.git \
##       to commit e7b75070c72a88f0f7926cc6872858a2c5f0090d
## Successfully built peft
#
#

channels:
  - defaults
dependencies:
  - python=3.10.14
  - pip=24.0
  - pip:
      - accelerate==0.30.1
      - bitsandbytes==0.43.1
      - datasets==2.19.1
      - evaluate==0.4.2
      - huggingface-hub==0.23.2
      - humanfriendly==10.0
      - jupyter==1.0.0
```

```
- nltk==3.8.1
- peft==0.11.2.dev0
- py-cpuinfo==9.0.0
- pylspci==0.4.3
- qwqfetch==0.0.0
- rouge-score==0.1.2
- tensorflow-cpu==2.16.1
- torch==2.3.0
- transformers==4.41.1
- trl==0.8.6
- wmi==1.5.1
```

Imports

In [15]:
```
from datasets import load_dataset
import random
from random import randrange
import torch
from transformers import AutoTokenizer, \
                         AutoModelForSeq2SeqLM, \
                         AutoModelForCausalLM, \
                         TrainingArguments, \
                         pipeline
from transformers.utils import logging
from peft import LoraConfig, \
                 prepare_model_for_kbit_training, \
                 get_peft_model, \
                 AutoPeftModelForCausalLM
from trl import SFTTrainer
from huggingface_hub import login, notebook_login

from datasets import load_metric
from evaluate import load as evaluate_dot_load
import nltk
import rouge_score
from rouge_score import rouge_scorer, scoring

import pickle
import pprint
import re
import timeit
from humanfriendly import format_timespan
import os

## my module(s), now just in the working directory as .PY files
import system_info_as_script
```

Load the training and test dataset along with the LLM with its tokenizer

The LLM will be fine-tuned. It seems the tokenizer will also be fine-tuned, but I'm not sure

Why aren't we loading the validation set? (I don't know; that's not a teaching question.)

I've tried to make use of it (the validation set) with the `trainer`. We'll see how it goes.

Update: It worked fine, though its loss is lower than the training set's loss.

In [67]:
```
#  Need to install  datasets  from pip, not conda. I'll do all from pip.
#+ I'll get rid of the current conda environment and make it anew.
#+ Actually, I'll make sure  conda  and  pip  are updated, then do what
#+ I discussed above.
#+
#+ cf.
#+     arch_ref_1 = "https://web.archive.org/web/20240522150357/" + \
#+                  "https://stackoverflow.com/questions/77433096/" + \
#+                  "notimplementederror-loading-a-dataset-" + \
#+                  "cached-in-a-localfilesystem-is-not-suppor"
#+
#+ Also useful might be
#+     arch_ref_2 = "https://web.archive.org/web/20240522150310/" + \
#+                  "https://stackoverflow.com/questions/76340743/" + \
#+                  "huggingface-load-datasets-gives-" + \
#+                  "notimplementederror-cannot-error"
#
data_files = {'train':'samsum-train.json',
              'evaluation':'samsum-validation.json',
              'test':'samsum-test.json'}
dataset = load_dataset('json', data_files=data_files)

model_name = "google/flan-t5-small"

model_load_tic = timeit.default_timer()
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
model_load_toc = timeit.default_timer()

model_load_duration = model_load_toc - model_load_tic

print(f"Loading the original model, {model_name}")
print(f"took {model_load_toc - model_load_tic:0.4f} seconds.")

model_load_time_str = format_timespan(model_load_duration)

print(f"which equates to {model_load_time_str}")

#  Next line makes training faster but a little less accurate
model.config.pretraining_tp = 1

tokenizer_tic = timeit.default_timer()
tokenizer = AutoTokenizer.from_pretrained(model_name,
                                          trust_remote_code=True)
tokenizer_toc = timeit.default_timer()

tokenizer_duration = tokenizer_toc - tokenizer_tic

print("Getting original tokenizer")
print(f"took {tokenizer_toc - tokenizer_tic:0.4f} seconds.")
```

```
tokenizer_time_str = format_timespan(tokenizer_duration)

print(f"which equates to {tokenizer_time_str}")

#  padding instructions for the tokenizer
#+   ??? !!! What about for RWKV !!! ???
#+ Will it be the same?
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
Loading the original model, google/flan-t5-small
took 0.9103 seconds.
which equates to 0.91 seconds
Getting original tokenizer
took 0.2560 seconds.
which equates to 0.26 seconds
```

Trying some things I've been learning

In [17]:
```
print(model)
```

```
T5ForConditionalGeneration(
  (shared): Embedding(32128, 512)
  (encoder): T5Stack(
    (embed_tokens): Embedding(32128, 512)
    (block): ModuleList(
      (0): T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): Linear(in_features=512, out_features=384, bias=False)
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): Linear(in_features=512, out_features=384, bias=False)
              (o): Linear(in_features=384, out_features=512, bias=False)
              (relative_attention_bias): Embedding(32, 6)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerFF(
            (DenseReluDense): T5DenseGatedActDense(
              (wi_0): Linear(in_features=512, out_features=1024, bias=False)
              (wi_1): Linear(in_features=512, out_features=1024, bias=False)
              (wo): Linear(in_features=1024, out_features=512, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
              (act): NewGELUActivation()
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
      (1-7): 7 x T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): Linear(in_features=512, out_features=384, bias=False)
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): Linear(in_features=512, out_features=384, bias=False)
              (o): Linear(in_features=384, out_features=512, bias=False)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerFF(
            (DenseReluDense): T5DenseGatedActDense(
              (wi_0): Linear(in_features=512, out_features=1024, bias=False)
              (wi_1): Linear(in_features=512, out_features=1024, bias=False)
              (wo): Linear(in_features=1024, out_features=512, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
              (act): NewGELUActivation()
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
```

```
    )
    (final_layer_norm): T5LayerNorm()
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (decoder): T5Stack(
    (embed_tokens): Embedding(32128, 512)
    (block): ModuleList(
      (0): T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): Linear(in_features=512, out_features=384, bias=False)
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): Linear(in_features=512, out_features=384, bias=False)
              (o): Linear(in_features=384, out_features=512, bias=False)
              (relative_attention_bias): Embedding(32, 6)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerCrossAttention(
            (EncDecAttention): T5Attention(
              (q): Linear(in_features=512, out_features=384, bias=False)
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): Linear(in_features=512, out_features=384, bias=False)
              (o): Linear(in_features=384, out_features=512, bias=False)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (2): T5LayerFF(
            (DenseReluDense): T5DenseGatedActDense(
              (wi_0): Linear(in_features=512, out_features=1024, bias=False)
              (wi_1): Linear(in_features=512, out_features=1024, bias=False)
              (wo): Linear(in_features=1024, out_features=512, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
              (act): NewGELUActivation()
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
      (1-7): 7 x T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): Linear(in_features=512, out_features=384, bias=False)
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): Linear(in_features=512, out_features=384, bias=False)
              (o): Linear(in_features=384, out_features=512, bias=False)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerCrossAttention(
```

```
            (EncDecAttention): T5Attention(
              (q): Linear(in_features=512, out_features=384, bias=False)
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): Linear(in_features=512, out_features=384, bias=False)
              (o): Linear(in_features=384, out_features=512, bias=False)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (2): T5LayerFF(
            (DenseReluDense): T5DenseGatedActDense(
              (wi_0): Linear(in_features=512, out_features=1024, bias=False)
              (wi_1): Linear(in_features=512, out_features=1024, bias=False)
              (wo): Linear(in_features=1024, out_features=512, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
              (act): NewGELUActivation()
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (final_layer_norm): T5LayerNorm()
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (lm_head): Linear(in_features=512, out_features=32128, bias=False)
)
```

In [18]:
```
model_arch_str = str(model)

with open("google_-flan-t5-small.model-architecture.txt", 'w', encoding='utf-8') as
    fh.write(model_arch_str)
##endof:  with open ... fh
```

In [19]:
```
# some other saves
pickle_filename = "lora_flan_t5_cpu_objects.pkl"
objects_to_pickle = []
objects_to_pickle.append(model_arch_str)
```

Prompt and Trainer

For our SFT (**S**upervised **F**ine **T**uning) model, we use the `class trl.SFTTrainer`.

I want to research this a bit, especially the `formatting_func` that we'll be passing to the `SFTTrainer`.

First, though, some information about SFT. From the Hugging Face Documentation at https://huggingface.co/docs/trl/en/sft_trainer (archived)

> Supervised fine-tuning (or SFT for short) is a crucial step in RLHF. In TRL we provide an easy-to-use API to create your SFT models and train them with few

> lines of code on your dataset.

Though I won't be using the examples unless I get even more stuck, the next paragraph *has* examples, and I'll put the paragraph here.

> Check out a complete flexible example at examples/scripts/sft.py [archived]. Experimental support for Vision Language Models is also included in the example examples/scripts/vsft_llava.py [archived].

RLHF (archived wikipedia page) is **R**einforcement **L**earning from **H**uman **F**eedback. TRL%20step.) (archived) **T**ransfer **R**einforcement **L**earning, a library from Hugging Face.

For the parameter, `formatting_func`, I can look ath the documentation site above (specifically here), at the GitHub repo for the code (in the docstrings), or from my local `conda` environment, at `C:\Users\bballdave025\.conda\envs\rwkv-lora-pat\Lib\site-packages\trl\trainer\sft_trainer.py`.

Pulling code from the last one, I get

```
    formatting_func (`Optional[Callable]`):
       The formatting function to be used for creating the
`ConstantLengthDataset`.
```

That matches the first very well

> **formatting_func** (`Optional[Callable]`) — The formatting function to be used for creating the `ConstantLengthDataset`.

(A quick note: In this Jupyter Notebook environment, I could have typed `trainer = SFTTrainer(` and then `Shift` + `Tab` to find that same documentation.

However, I think that more clarity is found at the documentation for `ConstantLengthDataset

> **formatting_func** (`Callable`, **optional**) — Function that formats the text before tokenization. Usually it is recommended to have follows a certain pattern such as `"### Question: {question} ### Answer: {answer}"`

So, as we'll see the next code from the tutorial, it basically is a prompt templater/formatter that matches the JSON. For example, we use `sample['dialogue']` to access the `dialogue` key/pair. That's what I got from all this stuff.

Mehul Gupta himself stated

> Next, using the Input and Output, we will create a prompt template which is a requirement by the SFTTrainer we will be using later

Prompt

In [68]:
```
def prompt_instruction_format(sample):
    return f""" Instruction:
      Use the Task below and the Input given to write the Response:

      ### Task:
      Summarize the Input

      ### Input:
      {sample['dialogue']}

      ### Response:
      {sample['summary']}
      """
##endof:  prompt_instruction_format(sample)
```

Trainer - the LoRA Setup Part

Arguments and Configuration

See this section to see what I changed from the tutorial to get the evaluation set as part of training and to get a customized repo name. The couple of sections before it will give more details.

In [69]:
```
#  Some arguments to pass to the trainer
training_args = TrainingArguments(
                        output_dir='output',
                        num_train_epochs=1,
                        per_device_train_batch_size=4,
                        save_strategy='epoch',
                        learning_rate=2e-4,
                        do_eval=True,
                        per_device_eval_batch_size=4,
                        eval_strategy='epoch',
                        hub_model_id="dwb-flan-t5-small-lora-finetune",
)

# the fine-tuning (peft for LoRA) stuff
peft_config = LoraConfig( lora_alpha=16,
                          lora_dropout=0.1,
                          r=64,
                          bias='none',
                          task_type='CAUSAL_LM'
)
```

`task_type` , cf. https://github.com/huggingface/peft/blob/main/src/peft/config.py#L222 (archived)

> ```
>    Args:
>        peft_type (Union[[`~peft.utils.config.PeftType`],
> `str`]): The type of Peft method to use.
>        task_type (Union[[`~peft.utils.config.TaskType`],
> `str`]): The type of task to perform.
> ```

```
        inference_mode (`bool`, defaults to `False`): Whether
    to use the Peft model in inference mode.
```

After some searching using Cygwin

```
bballdave025@MYMACHINE
/cygdrive/c/Users/bballdave025/.conda/envs/rwkv-lora-pat/Lib/site-
packages/peft/utils
$ ls -lah
total 116K
drwx------+ 1 bballdave025 bballdave025    0 May 28 21:09 .
drwx------+ 1 bballdave025 bballdave025    0 May 28 21:09 ..
-rwx------+ 1 bballdave025 bballdave025 2.0K May 28 21:09
__init__.py
drwx------+ 1 bballdave025 bballdave025    0 May 28 21:09
__pycache__
-rwx------+ 1 bballdave025 bballdave025 8.0K May 28 21:09
constants.py
-rwx------+ 1 bballdave025 bballdave025 3.8K May 28 21:09
integrations.py
-rwx------+ 1 bballdave025 bballdave025  17K May 28 21:09
loftq_utils.py
-rwx------+ 1 bballdave025 bballdave025 9.7K May 28 21:09
merge_utils.py
-rwx------+ 1 bballdave025 bballdave025  25K May 28 21:09 other.py
-rwx------+ 1 bballdave025 bballdave025 2.2K May 28 21:09
peft_types.py
-rwx------+ 1 bballdave025 bballdave025  21K May 28 21:09
save_and_load.py

bballdave025@MYMACHINE
/cygdrive/c/Users/bballdave025/.conda/envs/rwkv-lora-pat/Lib/site-
packages/peft/utils
$ grep -iIRHn "TaskType" .
peft_types.py:60:class TaskType(str, enum.Enum):
__init__.py:20:# from .config import PeftConfig, PeftType,
PromptLearningConfig, TaskType
__init__.py:22:from .peft_types import PeftType, TaskType

bballdave025@MYMACHINE
/cygdrive/c/Users/bballdave025/.conda/envs/rwkv-lora-pat/Lib/site-
packages/peft/utils
$
```

So, let's look at the `peft_types.py` file.

The docstring for `class TaskType(str, enum.Enum)` is

```
    Enum class for the different types of tasks supported by PEFT.

    Overview of the supported task types:
```

```
        - SEQ_CLS: Text classification.
        - SEQ_2_SEQ_LM: Sequence-to-sequence language modeling.
        - CAUSAL_LM: Causal language modeling.
        - TOKEN_CLS: Token classification.
        - QUESTION_ANS: Question answering.
        - FEATURE_EXTRACTION: Feature extraction. Provides the hidden
    states which can be used as embeddings or features
          for downstream tasks.
```

We're going to start timing stuff, so here's some system info

`system_info_as_script.py` is a script I wrote with the help of a variety of StackOverflow and documentation sources. It should be in the working directory.

In [ ]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y%m%dT%H%M%S%Z00\") -replace '[.][0-9]*_',
```

In [ ]:
```
system_info_as_script.run()
```

Before I rebooted (and I did reboot this morning by shutting down and restarting, though the script output it still shows the reboot date as `2024-5-26` and the uptime as `7 days, 18 hours, 53 minutes`), I ran this from an elevated command prompt. The result are in the file,

`system_info_win_compy_admin_2024-06-03T070700-0600.txt`

ROUGE Metrics

Some references from the Microsoft/Google (who?) implementation

https://pypi.org/project/rouge-score/

https://web.archive.org/web/20240530231357/https://pypi.org/project/rouge-score/

https://github.com/google-research/google-research/tree/master/rouge

https://web.archive.org/web/20240530231412/https://github.com/google-research/google-research/tree/master/rouge

Not the one I used:

https://github.com/microsoft/nlp-recipes/blob/master/examples/text_summarization/summarization_evaluation.ipynb

https://web.archive.org/web/20240530231709/https://github.com/microsoft/nlp-recipes/blob/master/examples/text_summarization/summarization_evaluation.ipynb

Someone else made this other one, which I inspected but didn't use.

https://pypi.org/project/rouge/

https://web.archive.org/web/20240530232029/https://pypi.org/project/rouge/

https://github.com/pltrdy/rouge

https://web.archive.org/web/20240530232023/https://github.com/pltrdy/rouge

but I think he defers to the rouge_score from Google.

My ROUGE Metrics

I want to use the skip-grams score. Thanks to

https://www.bomberbot.com/machine-learning/skip-bigrams-in-system/

https://web.archive.org/web/20240530230949/https://www.bomberbot.com/machine-learning/skip-bigrams-in-system/

I can do this as well as writing the code for the other metrics.

Not used for now

Focusing on the main goal. Quick and Reckless. My therapist would be so proud.

In [22]:
```
#import dwb_rouge_scores

#help(dwb_rouge_scores.dwb_rouge_n)

# print("SEPARATOR")

#help(dwb_rouge_scores.dwb_rouge_L)

# print("SMALLER-SEPARATOR\nwhich needs")

#help(dwb_rouge_scores.dwb_lcs)

# print("SEPARATOR")

#help(dwb_rouge_scores.dwb_rouge_s)

# print("SMALLER-SEPARATOR\nwhich needs")

#help(dwb_rouge_scores.dwb_skipngrams)

# print("SEPARATOR")
```

```
#help(dwb_rouge_scores.dwb_rouge_Lsum)

# print("which just wraps google-research's rouge_score's version")

#help()
```

Other useful ROUGE code

(found as I go along)

In [70]:
```
def format_rouge_score_rough(this_rouge_str,
                             do_debug_rouge_fmt=False):
    '''

    '''

    rouge_ret_str = this_rouge_str

    if do_debug_rouge_fmt:
        print(" #DEBUG 1#")
        print(rouge_ret_str)
    ##endof:  do_debug_rouge_fmt

    rouge_ret_str = re.sub(r"([(,][ ]?)([0-9A-Za-z_]+[=])",
                           "\g<1>\n     \g<2>",
                           rouge_ret_str,
                           flags=re.I|re.M
    )

    if do_debug_rouge_fmt:
        print(" #DEBUG 2#")
        print(rouge_ret_str)
    ##endof:  do_debug_rouge_fmt

    rouge_ret_str = re.sub(r"(.)([])])$",
                           "\g<1>\n\g<2>",
                           rouge_ret_str
    )

    if do_debug_rouge_fmt:
        print(" #DEBUG 3#")
        print(rouge_ret_str)
    ##endof:  do_debug_rouge_fmt

    rouge_ret_str = rouge_ret_str.replace(
                                  "precision=",
                                  "     precision="
                               ).replace(
                                  "recall=",
                                  "     recall="
                               ).replace(
                                  "fmeasure=",
                                  "     fmeasure="
    )
```

```
    return rouge_ret_str

##endof:  format_rouge_score_rough(<params>)
```

In [71]:
```
def print_rouge_scores(result, sample_num=None):
    '''

    '''

    print("\n\n---------- ROUGE SCORES ----------")
    if sample_num is None:
        print("  --------- dialogue ----------")
    elif type(sample_num) is int:
        print(f"  --------- dialogue {sample_num+1} ----------")
    else:
        print(f"  --------- {sample_num} ----------")
    ##endof:  if/else sample_num is None
    print("ROUGE-1 results")
    rouge1_str = str(result['rouge1'])
    print(format_rouge_score_rough(rouge1_str))
    print("ROUGE-2 results")
    rouge2_str = str(result['rouge2'])
    print(format_rouge_score_rough(rouge2_str))
    print("ROUGE-L results")
    rougeL_str = str(result['rougeL'])
    print(format_rouge_score_rough(rougeL_str))
    print("ROUGE-Lsum results")
    rougeLsum_str = str(result['rougeLsum'])
    print(format_rouge_score_rough(rougeLsum_str))
##endof:  print_rouge_scores(<params>)
```

In [72]:
```
#-------------------------------------------------------------------------------
# #  From https://github.com/google-research/google-research/tree/master/rouge
# #+ <strike>I can't see how to aggregate it, though I may have</strike>
# #+ I found a resource at
# #+  ref_gg_rg="https://github.com/huggingface/datasets/blob/" + \
# #+            "main/metrics/rouge/rouge.py"
# #+
# #+ arch_gg_rg="https://web.archive.org/web/20240603192938/" + \
# #+            "https://github.com/huggingface/datasets/blob/" + \
# #+            "main/metrics/rouge/rouge.py"
#
def compute_google_rouge_score(predictions,
                               references,
                               rouge_types=None,
                               use_aggregator=True,
                               use_stemmer=False):
    '''
    Figuring out the nice format of the deprecated method
    '''
    if rouge_types is None:
        rouge_types = ["rouge1", "rouge2", "rougeL", "rougeLsum"]
    ##endof:  if rouge_types is None
    scorer = rouge_scorer.RougeScorer(rouge_types=rouge_types,
```

```
                                        use_stemmer=use_stemmer
    )
    if use_aggregator:
        aggregator = scoring.BootstrapAggregator()
    else:
        scores = []
    ##endof:  if/else use_aggregator
    for ref, pred in zip(references, predictions):
        score = scorer.score(ref, pred)
        if use_aggregator:
            aggregator.add_scores(score)
        else:
            scores.append(score)
    ##endof:  for
    if use_aggregator:
        result = aggregator.aggregate()
    else:
        result = {}
        for key in scores[0]:
            result[key] = [score[key] for score in scores]
        ##endof:  for
    ##endof:  if/else
    return result
##endof:  compute_google_rouge_score
```

Extra cell.

Try for a baseline (for out-of-the-box, pretrained model)

Just one summarization to begin with, randomly picked

In [73]:
```
#  Just one summarization to begin with, randomly picked ... but
#+ now with th possibility of a known seed, to allow visual
#+ comparison with after-training results.
#+ I'M NOT GOING TO USE THIS REPEATED SEED, I'm just going to
#+ use the datum at the first index to compare.

do_seed_for_repeatable = True

summarizer = pipeline('summarization',
                      model=model,
                      tokenizer=tokenizer)

if do_seed_for_repeatable:
    rand_seed_for_randrange = 137
    random.seed(rand_seed_for_randrange)
##endof:  if do_seed_for_repeatable

sample = dataset['test'][randrange(len(dataset["test"]))]
print(f"dialogue: \n{sample['dialogue']}\n---------------")

res = summarizer(sample["dialogue"])

print(f"flan-t5-small summary:\n{res[0]['summary_text']}")
```

```
dialogue:
Jayden: But I don't need kids. Kids means over. At least for a woman
Brennan: Over what ?
Jayden: The end of normal life. Being pregnant, suffering because of this etc
Brennan: Hmm so I need to look for another mother to my kids then. Haha
Jayden: Being obligated to be with the. 24h. Men have only sex and they wait for kid
s while women suffer
Brennan: I don't agree...
Jayden: I wish I could do the same. Then probably i would say the same like u.
Brennan: Guys like me would be there through it all to reduce the suffering
Jayden: Physical suffering. No one can do anything with this. I wish I could just ha
ve sex and wait for a baby while having a normal life. Not getting fat, having the s
ame body, the same breast and not disgusting ... Not feeling sick, not having pain,
being able to do every day stuff even like walking...
Brennan: It's gonna happen eventually
Jayden: I was I'm a store, behind me there was a pregnant woman, she dropped some mo
ney and she couldn't even take them from the floor... I had to help her
Brennan: That's because she's about to give birth
Jayden: I hope that maybe soon they will be possible to have a child without being p
regnant. Yes! And she's suffering
Brennan: Any I'm sorry for feeding you with my bullshit
Jayden: While a man is doing his normal stuff. U mean the conversation?
Brennan: I hope you find a guy that can give you the sex you want and not get pregna
nt
Jayden: Would be awesome
Brennan: I'm gonna go to sleep now. Good night
Jayden: I said I don't want to have any children now! Maybe in the future when I hav
e a good job, I'm financially independent. Good night
---------------
flan-t5-small summary:
Jayden doesn't need kids. He needs to look for another mother to his kids. Jayden is
a store, behind him, and a pregnant woman dropped some money and couldn't take them
from the floor. She's about to give birth.
```

Now, a couple summarizations with comparison to ground truth

In [74]:
```
pred_test_list = []
ref_test_list = []

sample_num = 0

this_sample = dataset['test'][sample_num]

print(f"dialogue: \n{this_sample['dialogue']}\n---------------")

ground_summary = this_sample['summary']
res = summarizer(this_sample['dialogue'])
res_summary = res[0]['summary_text']

print(f"human-genratd summary:\n{ground_summary}")
print(f"flan-t5-small summary:\n{res_summary}")

ref_test_list.append(ground_summary)
pred_test_list.append(res_summary)

#-------------------------------------------------------------------------------
#  datasets.load_metric
#+ Supposed to be deprecated, but it's the only one I found that aggregates
#+ the scores. Also, it gives more than just an f-score
rouge = load_metric('rouge', trust_remote_code=True)

#  Yes, I have just one datum, but I'm setting things up to
#+ work well with a loop.
results_test_0 = rouge.compute(
                    predictions=pred_test_list,
                    references=ref_test_list,
                    use_aggregator=False
)

# >>> print(list(results_test.keys()))
# ['rouge1', 'rouge2', 'rougeL', 'rougeLsum']
```

```
Your max_length is set to 200, but your input_length is only 133. Since this is a su
mmarization task, where outputs shorter than the input are typically wanted, you mig
ht consider decreasing max_length manually, e.g. summarizer('...', max_length=66)
```

```
dialogue:
Hannah: Hey, do you have Betty's number?
Amanda: Lemme check
Hannah: <file_gif>
Amanda: Sorry, can't find it.
Amanda: Ask Larry
Amanda: He called her last time we were at the park together
Hannah: I don't know him well
Hannah: <file_gif>
Amanda: Don't be shy, he's very nice
Hannah: If you say so..
Hannah: I'd rather you texted him
Amanda: Just text him 🙂
Hannah: Urgh.. Alright
Hannah: Bye
Amanda: Bye bye
---------------
human-genratd summary:
Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry.
flan-t5-small summary:
Larry called Hannah last time she was at the park together. Hannah doesn't know Larr
y well. Larry called her last time they were at a park. Hannah will text Larry.
```

In [75]:
```
print_rouge_scores(results_test_0, 0)
```

```
---------- ROUGE SCORES ----------
  --------- dialogue 1 ----------
ROUGE-1 results
[Score(
          precision=0.16129032258064516,
          recall=0.3125,
          fmeasure=0.2127659574468085)]
ROUGE-2 results
[Score(
          precision=0.03333333333333333,
          recall=0.0666666666666667,
          fmeasure=0.04444444444444444)]
ROUGE-L results
[Score(
          precision=0.12903225806451613,
          recall=0.25,
          fmeasure=0.1702127659574468)]
ROUGE-Lsum results
[Score(
          precision=0.12903225806451613,
          recall=0.25,
          fmeasure=0.1702127659574468)]
```

In [76]:
```
sample_num = 224

this_sample = dataset['test'][sample_num]

print(f"dialogue: \n{this_sample['dialogue']}\n---------------")

ground_summary = this_sample['summary']
res = summarizer(this_sample['dialogue'])
res_summary = res[0]['summary_text']

print(f"human-genratd summary:\n{ground_summary}")
print(f"flan-t5-small summary:\n{res_summary}")

# Now, we'll have two data
ref_test_list = [ground_summary]
pred_test_list = [res_summary]

results_test_224 = rouge.compute(
                      predictions=pred_test_list,
                      references=ref_test_list,
                      use_aggregator=False
)
```

```
Your max_length is set to 200, but your input_length is only 160. Since this is a su
mmarization task, where outputs shorter than the input are typically wanted, you mig
ht consider decreasing max_length manually, e.g. summarizer('...', max_length=80)
dialogue:
Abigail: It's Sundaay.
Damien: So?..
Abigail: You know what that means.
Damien: Hmm no I don't x)
Abigail: Sunday means we go to church~.
Damien: Oh, yeah..
Abigail: Don't forget to put on a coat and tie.
Damien: A coat and tie?.. Why?
Abigail: To show respect to God and others.
Damien: Omg..I'm glad Sunday is only once a week.
Abigail: I hope God didn't hear that.
Damien: He'll forgive me 😇
Abigail: Just be ready on time please.
---------------
human-genratd summary:
Abigail and Damien are going to church on Sunday. Damien has to put on a coat and ti
e.
flan-t5-small summary:
Abigail, Damien and Damien go to church on Sunday. They are going to pray for God an
d others. Damien is glad Sunday is only once a week.
```

In [77]:
```
print_rouge_scores(results_test_224, 224)
```

```
---------- ROUGE SCORES ----------
  --------- dialogue 225 ----------
ROUGE-1 results
[Score(
          precision=0.48148148148148145,
          recall=0.7222222222222222,
          fmeasure=0.5777777777777777)]
ROUGE-2 results
[Score(
          precision=0.23076923076923078,
          recall=0.35294117647058826,
          fmeasure=0.2790697674418605)]
ROUGE-L results
[Score(
          precision=0.3333333333333333,
          recall=0.5,
          fmeasure=0.4)]
ROUGE-Lsum results
[Score(
          precision=0.3333333333333333,
          recall=0.5,
          fmeasure=0.4)]
```

Note on ROUGE Scores

```
# @todo : Run the ROUGE analysis from the Python package
#         (after running with  trust_remote_code=False
#          to find the deprecation it mentioned).

#-----------------------------------------------------------------
------------
# #  From https://github.com/google-research/google-
research/tree/master/rouge
# #+ I can't see how to aggregate it, though I may have found a
resource at
# #+  ref_gg_rg="https://github.com/huggingface/datasets/blob/" + \
# #+            "main/metrics/rouge/rouge.py"
# #+
# #+ arch_gg_rg="https://web.archive.org/web/20240603192938/" + \
# #+            "https://github.com/huggingface/datasets/blob/" + \
# #+            "main/metrics/rouge/rouge.py"
#

#  It turns out that the deprecated one is preferable in
#+ output, at least until I can debug the aggregation of
#+ scores with another version: compute_google_rouge_score
```

That should come from the `compute_google_rouge_score`, above. I was able to look through the code for `datasets.load_metric('rouge')` code and put together that method.

For now, I used ...

```
# Using the deprecated-but-aggregating-and-not-only-f-score one
rouge = load_metric('rouge', trust_remote_code=False)
```

This next one is what the warning message said to use, but it only returns an f-measure (f-score)

```
# #  Replacement for the load_metric - evaluate.load(metric_name)
# #+ Docs said:
# #+
# #+> Returns:
# #+>    rouge1: rouge_1 (f1),
# #+>    rouge2: rouge_2 (f1),
```

```
    # #+>    rougeL: rouge_l (f1),
    # #+>    rougeLsum: rouge_lsum (f1)
    # #+>
    # #+> Meaning we only get the f-score. I want more to compare.
    # #-v- code
    # rouge = evaluate_dot_load('rouge')
```

Verbosity stuff - get rid of the nice advice

In [36]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y%m%dT%H%M%S%Z00\") -replace '[.][0-9]*_',
```

```
1717519975_20240604T165255-0600
```

Output was:

```
1717411179_20240603T103939-0600
```

In [37]:
```
log_verbosity_is_critical = \
  logging.get_verbosity() == logging.CRITICAL # alias FATAL, 50
log_verbosity_is_error = \
  logging.get_verbosity() == logging.ERROR # 40
log_verbosity_is_warn = \
  logging.get_verbosity() == logging.WARNING # alias WARN, 30
log_verbosity_is_info = \
  logging.get_verbosity() == logging.INFO # 20
log_verbosity_is_debug = \
  logging.get_verbosity() == logging.DEBUG # 10

print( "The statement, 'logging verbosity is CRITICAL' " + \
      f"is {log_verbosity_is_critical}")
print( "The statement, 'logging verbosity is    ERROR' " + \
      f"is {log_verbosity_is_error}")
print( "The statement, 'logging verbosity is  WARNING' " + \
      f"is {log_verbosity_is_warn}")
print( "The statement, 'logging verbosity is     INFO' " + \
      f"is {log_verbosity_is_info}")
print( "The statement, 'logging verbosity is    DEBUG' " + \
      f"is {log_verbosity_is_debug}")

print()

init_log_verbosity = logging.get_verbosity()
print(f"The value of logging.get_verbosity() is: {init_log_verbosity}")

print()

init_t_n_a_w = os.environ.get('TRANSFORMERS_NO_ADVISORY_WARNINGS')
print(f"TRANSFORMERS_NO_ADIVSORY_WARNINGS: {init_t_n_a_w}")
```

```
The statement, 'logging verbosity is CRITICAL' is False
The statement, 'logging verbosity is    ERROR' is False
The statement, 'logging verbosity is  WARNING' is True
The statement, 'logging verbosity is     INFO' is False
The statement, 'logging verbosity is    DEBUG' is False

The value of logging.get_verbosity() is: 30

TRANSFORMERS_NO_ADIVSORY_WARNINGS: None
```

Actual Baseline

In [38]:
```
# # Don't need this again
# !powershell -c (Get-Date -UFormat \"%s_%Y%m%dT%H%M%S%Z00\") -replace '[.][0-9]*_'
```

Output was:

```
1717411242_20240603T104042-0600
```

!!! NOTE You'd better **make dang sure you want the lots of output** before you set this next boolean to `True`

In [78]:
```
do_have_lotta_output_from_all_dialogs_summaries_1 = False
```

Are you sure about the value of that last boolean? 1

There could be megabytes (maybe gigabytes) worth of text output if you've changed it to `True`.

Actual Baseline

In [40]:
```
#  ref1 = "https://web.archive.org/web/20240530051418/" + \
#+        "https://stackoverflow.com/questions/73221277/" + \
#+        "python-hugging-face-warning"
#  ref2 = "https://web.archive.org/web/20240530051559/" + \
#+        "https://huggingface.co/docs/transformers/en/" + \
#+        "main_classes/logging"

##  Haven't tried this, because the logging seemed easier,
##+ and the logging worked
#os.environ("TRANSFORMERS_NO_ADVISORY_WARNINGS") = 1

logging.set_verbosity_error()

summarizer = pipeline('summarization',
                      model=model,
                      tokenizer=tokenizer)

#*p*#baseline_sample_dialog_list = []
baseline_prediction_list = []
baseline_reference_list = []

baseline_tic = timeit.default_timer()

for sample_num in range(len(dataset['test'])):
    this_sample = dataset['test'][sample_num]

    if do_have_lotta_output_from_all_dialogs_summaries_1:
        print(f"dialogue: \n{this_sample['dialogue']}\n---------------")
    ##endof:  if do_have_lotta_output_from_all_dialogs_summaries_1

    ground_summary = this_sample['summary']
    res = summarizer(this_sample['dialogue'])
    res_summary = res[0]['summary_text']

    if do_have_lotta_output_from_all_dialogs_summaries_1:
        print(f"human-genratd summary:\n{ground_summary}")
        print(f"flan-t5-small summary:\n{res_summary}")
    ##endof:  if do_have_lotta_output_from_all_dialogs_summaries_1

    #*p*#    baseline_sample_dialog_list.append(this_sample)
    baseline_reference_list.append(ground_summary)
    baseline_prediction_list.append(res_summary)
##endof:  for sample_num in range(len(dataset['test']))

baseline_toc = timeit.default_timer()

baseline_duration = baseline_toc - baseline_tic

print( "Getting things ready for scoring")
print(f"took {baseline_toc - baseline_tic:0.4f} seconds.")

#  It turns out that the deprecated one is preferable in
```

```
#+ output, at least until I can debug the aggregation of
#+ scores with another version
#+ That should come with the

rouge = load_metric('rouge', trust_remote_code=False)

baseline_results = rouge.compute(
                      predictions=baseline_prediction_list,
                      references=baseline_reference_list,
                      use_aggregator=True
)

# >>> print(list(baseline_results.keys()))
# ['rouge1', 'rouge2', 'rougeL', 'rougeLsum']

#*p*# objects_to_pickle.append(baseline_sample_dialog_list)
#*p*# objects_to_pickle.append(baseline_prediction_list)
#*p*# objects_to_pickle.append(baseline_reference_list)
#*p*# objects_to_pickle.append(baseline_results)
```

```
Getting things ready for scoring
took 1113.8523 seconds.
```

In [43]:
```
print_rouge_scores(baseline_results)
```

```
---------- ROUGE SCORES ----------
  --------- dialogue ----------
ROUGE-1 results
AggregateScore(
     low=Score(
          precision=0.36320630445704477,
          recall=0.5391471908229872,
          fmeasure=0.41209971865595346),
     mid=Score(
          precision=0.37394711195774655,
          recall=0.5518956018541074,
          fmeasure=0.4216852406490635),
     high=Score(
          precision=0.3843089278286546,
          recall=0.5652673531194096,
          fmeasure=0.43106509690207256)
)
ROUGE-2 results
AggregateScore(
     low=Score(
          precision=0.15921598436893325,
          recall=0.24399260896723063,
          fmeasure=0.18098064580068599),
     mid=Score(
          precision=0.16751331807822,
          recall=0.25688418792453044,
          fmeasure=0.1901013569791662),
     high=Score(
          precision=0.17601669526453642,
          recall=0.26996925142296735,
          fmeasure=0.1988747178644448)
)
ROUGE-L results
AggregateScore(
     low=Score(
          precision=0.2798170544171966,
          recall=0.4220715282711129,
          fmeasure=0.31929134202126586),
     mid=Score(
          precision=0.28896822314514115,
          recall=0.43511544077895614,
          fmeasure=0.32786822093032963),
     high=Score(
          precision=0.29854357582265284,
          recall=0.44899752808600696,
          fmeasure=0.33655474992458917)
)
ROUGE-Lsum results
AggregateScore(
     low=Score(
          precision=0.2803262832798807,
          recall=0.4225291787351153,
          fmeasure=0.31968927668471403),
     mid=Score(
          precision=0.28924184457875435,
```

```
                    recall=0.4348222878968877,
                    fmeasure=0.3278854406001706),
               high=Score(
                    precision=0.2986060650799353,
                    recall=0.4471497194451444,
                    fmeasure=0.3366741267731763)
          )
```

In [79]:
```
##  Haven't tried this, because the logging seemed easier,
##+ and the logging worked
# os.environ("TRANSFORMERS_NO_ADVISORY_WARNINGS") = init_t_n_a_w

logging.set_verbosity(init_log_verbosity)
```

In [45]:
```
do_enter_duration_manually = False
NUM_TO_CATCH_NO_MANUAL_ENTRY = -137.
is_a_manual_entry_skip = False # innocent until proven guilty


if do_enter_duration_manually:
    # !!! remember to type in your number, if needed !!! #
    baseline_duration = NUM_TO_CATCH_NO_MANUAL_ENTRY
    # !!! UNCOMMENT THE NEXT LINE IF YOU WANT TO ENTER MANUALLY !!!
    #baseline_duration = 1162.5236
##endof:  if do_enter_duration_manually

print("Running baseline inference (using the test set)")
```

```
if ( ( do_enter_duration_manually ) and \
     ( baseline_duration == -137. ) \
   ):
    print("took AN UNKNOWN AMOUNT OF TIME.")
    print("You didn't manually enter in your real time,")
    print("as you should have.")
    is_a_manual_entry_skip = True
elif ( ( do_enter_duration_manually ) and \
       ( baseline_duration != -137. )
     ):
    print("(and using your manually entered time)")
else:
    pass
##endof:  if <check manual entry>

if not is_a_manual_entry_skip:
    print(f"took {format_timespan(baseline_duration)}")
##endof:  if not is_a_manual_entry_skip
```

```
Running baseline inference (using the test set)
took 18 minutes and 33.85 seconds
```

Trainer - the Actual Trainer Part

In [ ]:
```
# # Don't need this again
# !powershell -c (Get-Date -UFormat \"%s_%Y-%m-%dT%H%M%S%Z00\") -replace '[.][0-9]*
```

Output was:

```
1717096214_2024-05-30T191014-0600
```

In [ ]:
```
trainer = SFTTrainer( model=model,
                      train_dataset=dataset['train'],
                      eval_dataset=dataset['evaluation'],
                      peft_config=peft_config,
                      tokenizer=tokenizer,
                      packing=True,
                      formatting_func=prompt_instruction_format,
                      args=training_args,
                    )
##  Warnings are below output.

##  Ended up not using this.
#                       max_seq_length=675
#          )
```

First time warnings from the code above (as it still is).

```
   WARNING:bitsandbytes.cextension:The installed version of
bitsandbytes \
    was compiled without GPU support. 8-bit optimizers, 8-bit
multiplication, \
    and GPU quantization are unavailable.
   C:\Users\bballdave025\.conda\envs\rwkv-lora-pat\lib\site-
```

```
packages\trl\\
    trainer\sft_trainer.py:246: UserWarning: You didn't pass a
`max_seq_length` \
   argument to the SFTTrainer, this will default to 512
    warnings.warn(

   [ > Generating train split: 6143/0 [00:04<00:00, 2034.36
examples/s] ]

   Token indices sequence length is longer than the specified
maximum sequence \
    length for this model (657 > 512). Running this sequence
through the model \
    will result in indexing errors

   [ > Generating train split: 355/0 [00:00<00:00, 6.10
examples/s] ]
```

DWB Note and possible

@todo:

~~So, I'm changing the `max_seq_length`~~ : Maybe I should just throw out the offender(s) (along with the blank one that's in there somewhere), but I'll just continue as is.

Actually, it appears I didn't run the updated cell, (with `max_seq_length=675`), since the Warning and Advice are still there.

Let's Train This LoRA Thing and See How It Does!

In [ ]:
```
# # Don't need this again
# !powershell -c (Get-Date -UFormat \"%s_%Y-%m-%dT%H%M%S%Z00\") -replace '[.][0-9]*
```

Output was:

`1717096271_2024-05-30T191111-0600`

At about `1717063394_2024-05-30T100314-0600`, DWB went in and renamed `profile.ps1` to `NOT-USING_-_pro_file_-_now.ps1.bak` That should get rid of our errors from `powershell`

The long-time-taking training code is just below.

In [ ]:
```
tic = timeit.default_timer()
trainer.train()
toc = timeit.default_timer()
print(f"tic: {tic}")
```

```
print(f"toc: {toc}")
training_duration = toc - tic
print(f"Training took {toc - tic:0.4f} seconds.")
```

In [ ]:
```
do_by_hand = False
NUM_TO_CATCH_NO_DO_BY_HAND = -137.
is_a_do_by_hand_skip = False # innocent until proven guilty


if do_by_hand:
    # !!! remember to type in your number, if needed !!! #
    training_duration = NUM_TO_CATCH_NO_MANUAL_ENTRY
    # !!! UNCOMMENT THE NEXT LINE IF YOU WANT TO ENTER MANUALLY !!!
    #training_duration = 11081.7024
##endof:  if do_by_hand

print("Running training with LoRA")
print("(using the training and eval sets)")
if ( ( do_by_hand ) and \
     ( training_duration == -137. ) \
   ):
    print("took AN UNKNOWN AMOUNT OF TIME.")
    print("You didn't manually enter in your real time,")
    print("as you should have.")
    is_a_do_by_hand_skip = True
elif ( ( do_by_hand ) and \
       ( training_duration != -137. )
     ):
    print("(and using your manually entered time)")
else:
    pass
##endof:  if <check manual entry>

if not is_a_do_by_hand_skip:
    print(f"took {format_timespan(training_duration)}")
##endof:  if not is_a_do_by_hand_skip
```

In [ ]:
```
# # Don't need this again
# !powershell -c (Get-Date -UFormat \"%s_%Y-%m-%dT%H%M%S%Z00\") -replace '[.][0-9]*
```

Output was:

```
1717107458_2024-05-30T221738-0600
```

@todo : consolidate "the other info as above"

I'm talking about the numbers of data points, tokens, whatever.

Any Comments / Things to Try (?)

We passed an evaluation set (parameter ``) to the `trainer`. How can we see information about that?

How to get the evaluation set used by the trainer

I added the following parameters to the `training_args = TrainingArguments(<args>)` call.

- `do_eval=True`
- `per_device_eval_batch_size=4`
- `eval_strategy='epoch'`

How to specify your repo name

I also added this next parameter to the arguments for `training_args = TrainingArguments(<args>)`

- `hub_model_id="dwb-flan-t5-small-lora-finetune"`

The final TrainingArguments call - with parameter list

```
training_args = TrainingArguments(
                        output_dir='output',
                        num_train_epochs=1,
                        per_device_train_batch_size=4,
                        save_strategy='epoch',
                        learning_rate=2e-4,
                        do_eval=True,
                        per_device_eval_batch_size=4,
                        eval_strategy='epoch',
                        hub_model_id="dwb-flan-t5-small-lora-
finetune",
)
```

Save the Trainer to Hugging Face and Get Our Updated Model

In [ ]:
```
# # Don't need this again
# !powershell -c (Get-Date -UFormat \"%s_%Y-%m-%dT%H%M%S%Z00\") -replace '[.][0-9]*
```

Output was:

`1717145367_2024-05-31T084927-0600`

I'm following the (archived) tutorial from Mehul Gupta on Medium; since it's archived, you can follow exactly what I'm doing.

Running this next line of code will come up with a dialog box with text entry, and I'm now using the `@thebballdave025` for Hugging Face stuff.

Make sure to use the WRITE token, here.

In [ ]:
```
#  This will come up with a dialog box with text entry.
#+ and I'm now using @thebballdave025 for Hugging Face.

# Use the write token, here.
notebook_login()
```

In [ ]:
```
# Save tokenizer and create a tokenizer model card
tokenizer.save_pretrained('testing')
  #  used 'testing' first - I think I can make a repo according
  #+ to the first getting-started cli instructions, but let's
  #+ use what Mehul Gupta used, first
  #  Actually, I think 'testing' is the local directory

# Create the trainer model card
trainer.create_model_card()

# Push the results to the Hugging Face Hub
trainer.push_to_hub()
```

Part of the output included the URL,

https://huggingface.co/thebballdave025/dwb-flan-t5-small-lora-finetune/commit/c87d34b398f3801ceb1e18c819a7c8fc894989c7

Hooray! The repo name I used in constructing the trainer worked!

I can get to the general repo with the URL,

https://huggingface.co/thebballdave025/dwb-flan-t5-small-lora-finetune

---

Info on the Fine-Tuned Model from the Repo's README - Model Card(?)

thebballdave025/dwb-flan-t5-small-lora-finetune

[archived] The archiving attempt at archive.org (Wayback Machine) failed. I'm not sure why, as the model is set as public.

```
PEFT  TensorBoard  Safetensors       generator  trl  sft
generated_from_trainer       License: apache-2.0
```

@todo : Edit Model Card

Unable to determine this model's pipeline type. Check the docs (i).

Adapter for google/flan-t5-small

dwb-flan-t5-small-lora-finetune

This model is a fine-tuned version of google/flan-t5-small on the generator dataset [DWB note: I don't know why it says "generator dataset". I used the samsum dataset, which I will link here and on the model card, eventually].

It achieves the following results on the evaluation set:

- Loss: 0.0226
- *DWB Note: I don't know which metric was used to calculate loss. If this were more important, I'd dig through code to find out and evaluate with the same metric. If I'm really lucky, they somehow used the ROUGE scores in the loss function, so we match.*

Model description

More information needed

Intended uses & limitations

More information needed

Training and evaluation data

More information needed

Training procedure

Training hyperparameters

The following hyperparameters were used during training:

- learning_rate: 0.0002
- train_batch_size: 4
- eval_batch_size: 4
- seed: 42
- optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
- lr_scheduler_type: linear
- num_epochs: 1

Training results

```
Training Loss | Epoch | Step | Validation Loss
```

```
---------------+-------+------+-----------------
     0.0685    |  1.0  | 1536 |     0.0226
```

Framework versions

- PEFT 0.11.2.dev0
- Transformers 4.41.1
- Pytorch 2.3.0+cpu
- Datasets 2.19.1
- Tokenizers 0.19.1

Actually Get the Model from Hugging Face

Running this next line of code will come up with a dialog box with text entry, and I'm now using the `@thebballdave025` for Hugging Face stuff.

Make sure to use the READ token, here.

In [5]:
```
# Read token. Will bring up text entry to paste token string
notebook_login()
```

```
VBox(children=(HTML(value='<center> <img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…
```

In [6]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y-%m-%dT%H%M%S%Z00\") -replace '[.][0-9]*_'
```

```
1717491686_2024-06-04T090126-0600
```

Output was:

`1717491686_2024-06-04T090126-0600`

I have restarted the kernel since defining these.

In [80]:
```
#data_files = {'train':'samsum-train.json',
#               'evaluation':'samsum-validation.json',
#               'test':'samsum-test.json'}
#dataset = load_dataset('json', data_files=data_files)
```

In [81]:
```
# My trained model from Hugging Face

new_model_name = "thebballdave025/dwb-flan-t5-small-lora-finetune"
```

In [92]:
```
new_model_load_tic = timeit.default_timer()
# do we need
new_model = AutoModelForSeq2SeqLM.from_pretrained(new_model_name)
# # or, do we need
# new_model = AutoModelForCausalLM.from_pretrained(new_model_name)
new_model_load_toc = timeit.default_timer()
```

```
new_model_load_duration = new_model_load_toc - new_model_load_tic

print(f"Loading the LoRA-fine-tuned model, {new_model_name}")
print(f"took {new_model_load_toc - new_model_load_tic:0.4f} seconds.")

new_model_load_time_str = format_timespan(new_model_load_duration)

print(f"which equates to {new_model_load_time_str}")

#  Next line makes training faster but a little less accurate
new_model.config.pretraining_tp = 1

new_tokenizer_tic = timeit.default_timer()
new_tokenizer = AutoTokenizer.from_pretrained(new_model_name,
                                              trust_remote_code=True)
new_tokenizer_toc = timeit.default_timer()

new_tokenizer_duration = new_tokenizer_toc - new_tokenizer_tic

print()
print("Getting fine-turned tokenizer")
print(f"took {new_tokenizer_toc - new_tokenizer_tic:0.4f} seconds.")

new_tokenizer_time_str = format_timespan(new_tokenizer_duration)

print(f"which equates to {new_tokenizer_time_str}")

#  padding instructions for the tokenizer
#+   ??? !!! What about for RWKV !!! ???
#+ Will it be the same?
new_tokenizer.pad_token = new_tokenizer.eos_token
new_tokenizer.padding_side = "right"

#-----------------------------------------------------
# Got some weird results, so I'm doing the old tokenizer
old_model_name = "google/flan-t5-small"

##old_model_load_tic = timeit.default_timer()
##old_model = \
##     AutoModelForSeq2SeqLM.from_pretrained(old_model_name)
##old_model_load_toc = timeit.default_timer()

##old_model_load_duration = \
##           old_model_load_toc - old_model_load_tic

##print(f"Loading the original model, {old_model_name}")
##print("took " + \
##      f"{old_model_load_toc - old_model_load_tic:0.4f}" + \
##      " seconds."
##)

##old_model_load_time_str = format_timespan(old_model_load_duration)

##print(f"which equates to {old_model_load_time_str}")
```

```
## #  Next line makes training faster but a little less accurate
##old_model.config.pretraining_tp = 1

old_tokenizer_tic = timeit.default_timer()
old_tokenizer = AutoTokenizer.from_pretrained(
                                        old_model_name,
                                        trust_remote_code=True
)
old_tokenizer_toc = timeit.default_timer()

old_tokenizer_duration = old_tokenizer_toc - old_tokenizer_tic

print("Getting old tokenizer")
print( "took " + \
      f"{old_tokenizer_toc - old_tokenizer_tic:0.4f}"
       " seconds."
)

tokenizer_time_str = format_timespan(tokenizer_duration)

print(f"which equates to {tokenizer_time_str}")

#  padding instructions for the tokenizer
#+   ??? !!! What about for RWKV !!! ???
#+ Will it be the same?
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
C:\Users\Anast\.conda\envs\rwkv-lora-pat\lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you want to force a new download, use `force_download=True`.
  warnings.warn(
Loading the LoRA-fine-tuned model, thebballdave025/dwb-flan-t5-small-lora-finetune
took 1.6837 seconds.
which equates to 1.68 seconds

Getting fine-turned tokenizer
took 0.2179 seconds.
which equates to 0.22 seconds
Getting old tokenizer
took 0.4262 seconds.
which equates to 0.26 seconds
```

In [83]:
```
print(new_model)
```

```
T5ForConditionalGeneration(
  (shared): Embedding(32128, 512)
  (encoder): T5Stack(
    (embed_tokens): Embedding(32128, 512)
    (block): ModuleList(
      (0): T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): lora.Linear(
                (base_layer): Linear(in_features=512, out_features=384, bias=False)
                (lora_dropout): ModuleDict(
                  (default): Dropout(p=0.1, inplace=False)
                )
                (lora_A): ModuleDict(
                  (default): Linear(in_features=512, out_features=64, bias=False)
                )
                (lora_B): ModuleDict(
                  (default): Linear(in_features=64, out_features=384, bias=False)
                )
                (lora_embedding_A): ParameterDict()
                (lora_embedding_B): ParameterDict()
              )
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): lora.Linear(
                (base_layer): Linear(in_features=512, out_features=384, bias=False)
                (lora_dropout): ModuleDict(
                  (default): Dropout(p=0.1, inplace=False)
                )
                (lora_A): ModuleDict(
                  (default): Linear(in_features=512, out_features=64, bias=False)
                )
                (lora_B): ModuleDict(
                  (default): Linear(in_features=64, out_features=384, bias=False)
                )
                (lora_embedding_A): ParameterDict()
                (lora_embedding_B): ParameterDict()
              )
              (o): Linear(in_features=384, out_features=512, bias=False)
              (relative_attention_bias): Embedding(32, 6)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerFF(
            (DenseReluDense): T5DenseGatedActDense(
              (wi_0): Linear(in_features=512, out_features=1024, bias=False)
              (wi_1): Linear(in_features=512, out_features=1024, bias=False)
              (wo): Linear(in_features=1024, out_features=512, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
              (act): NewGELUActivation()
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
```

```
      )
      (1-7): 7 x T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): lora.Linear(
                (base_layer): Linear(in_features=512, out_features=384, bias=False)
                (lora_dropout): ModuleDict(
                  (default): Dropout(p=0.1, inplace=False)
                )
                (lora_A): ModuleDict(
                  (default): Linear(in_features=512, out_features=64, bias=False)
                )
                (lora_B): ModuleDict(
                  (default): Linear(in_features=64, out_features=384, bias=False)
                )
                (lora_embedding_A): ParameterDict()
                (lora_embedding_B): ParameterDict()
              )
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): lora.Linear(
                (base_layer): Linear(in_features=512, out_features=384, bias=False)
                (lora_dropout): ModuleDict(
                  (default): Dropout(p=0.1, inplace=False)
                )
                (lora_A): ModuleDict(
                  (default): Linear(in_features=512, out_features=64, bias=False)
                )
                (lora_B): ModuleDict(
                  (default): Linear(in_features=64, out_features=384, bias=False)
                )
                (lora_embedding_A): ParameterDict()
                (lora_embedding_B): ParameterDict()
              )
              (o): Linear(in_features=384, out_features=512, bias=False)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerFF(
            (DenseReluDense): T5DenseGatedActDense(
              (wi_0): Linear(in_features=512, out_features=1024, bias=False)
              (wi_1): Linear(in_features=512, out_features=1024, bias=False)
              (wo): Linear(in_features=1024, out_features=512, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
              (act): NewGELUActivation()
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (final_layer_norm): T5LayerNorm()
    (dropout): Dropout(p=0.1, inplace=False)
  )
```

```
    (decoder): T5Stack(
      (embed_tokens): Embedding(32128, 512)
      (block): ModuleList(
        (0): T5Block(
          (layer): ModuleList(
            (0): T5LayerSelfAttention(
              (SelfAttention): T5Attention(
                (q): lora.Linear(
                  (base_layer): Linear(in_features=512, out_features=384, bias=False)
                  (lora_dropout): ModuleDict(
                    (default): Dropout(p=0.1, inplace=False)
                  )
                  (lora_A): ModuleDict(
                    (default): Linear(in_features=512, out_features=64, bias=False)
                  )
                  (lora_B): ModuleDict(
                    (default): Linear(in_features=64, out_features=384, bias=False)
                  )
                  (lora_embedding_A): ParameterDict()
                  (lora_embedding_B): ParameterDict()
                )
                (k): Linear(in_features=512, out_features=384, bias=False)
                (v): lora.Linear(
                  (base_layer): Linear(in_features=512, out_features=384, bias=False)
                  (lora_dropout): ModuleDict(
                    (default): Dropout(p=0.1, inplace=False)
                  )
                  (lora_A): ModuleDict(
                    (default): Linear(in_features=512, out_features=64, bias=False)
                  )
                  (lora_B): ModuleDict(
                    (default): Linear(in_features=64, out_features=384, bias=False)
                  )
                  (lora_embedding_A): ParameterDict()
                  (lora_embedding_B): ParameterDict()
                )
                (o): Linear(in_features=384, out_features=512, bias=False)
                (relative_attention_bias): Embedding(32, 6)
              )
              (layer_norm): T5LayerNorm()
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (1): T5LayerCrossAttention(
              (EncDecAttention): T5Attention(
                (q): lora.Linear(
                  (base_layer): Linear(in_features=512, out_features=384, bias=False)
                  (lora_dropout): ModuleDict(
                    (default): Dropout(p=0.1, inplace=False)
                  )
                  (lora_A): ModuleDict(
                    (default): Linear(in_features=512, out_features=64, bias=False)
                  )
                  (lora_B): ModuleDict(
                    (default): Linear(in_features=64, out_features=384, bias=False)
                  )
                  (lora_embedding_A): ParameterDict()
```

```
          (lora_embedding_B): ParameterDict()
        )
        (k): Linear(in_features=512, out_features=384, bias=False)
        (v): lora.Linear(
          (base_layer): Linear(in_features=512, out_features=384, bias=False)
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.1, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=512, out_features=64, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=64, out_features=384, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (o): Linear(in_features=384, out_features=512, bias=False)
      )
      (layer_norm): T5LayerNorm()
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (2): T5LayerFF(
      (DenseReluDense): T5DenseGatedActDense(
        (wi_0): Linear(in_features=512, out_features=1024, bias=False)
        (wi_1): Linear(in_features=512, out_features=1024, bias=False)
        (wo): Linear(in_features=1024, out_features=512, bias=False)
        (dropout): Dropout(p=0.1, inplace=False)
        (act): NewGELUActivation()
      )
      (layer_norm): T5LayerNorm()
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
)
(1-7): 7 x T5Block(
  (layer): ModuleList(
    (0): T5LayerSelfAttention(
      (SelfAttention): T5Attention(
        (q): lora.Linear(
          (base_layer): Linear(in_features=512, out_features=384, bias=False)
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.1, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=512, out_features=64, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=64, out_features=384, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
        (k): Linear(in_features=512, out_features=384, bias=False)
        (v): lora.Linear(
          (base_layer): Linear(in_features=512, out_features=384, bias=False)
```

```
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.1, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=512, out_features=64, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=64, out_features=384, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (o): Linear(in_features=384, out_features=512, bias=False)
    )
    (layer_norm): T5LayerNorm()
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (1): T5LayerCrossAttention(
    (EncDecAttention): T5Attention(
      (q): lora.Linear(
        (base_layer): Linear(in_features=512, out_features=384, bias=False)
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.1, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=512, out_features=64, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=64, out_features=384, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (k): Linear(in_features=512, out_features=384, bias=False)
      (v): lora.Linear(
        (base_layer): Linear(in_features=512, out_features=384, bias=False)
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.1, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=512, out_features=64, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=64, out_features=384, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (o): Linear(in_features=384, out_features=512, bias=False)
    )
    (layer_norm): T5LayerNorm()
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (2): T5LayerFF(
    (DenseReluDense): T5DenseGatedActDense(
      (wi_0): Linear(in_features=512, out_features=1024, bias=False)
```

```
            (wi_1): Linear(in_features=512, out_features=1024, bias=False)
            (wo): Linear(in_features=1024, out_features=512, bias=False)
            (dropout): Dropout(p=0.1, inplace=False)
            (act): NewGELUActivation()
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (final_layer_norm): T5LayerNorm()
  (dropout): Dropout(p=0.1, inplace=False)
 )
 (lm_head): Linear(in_features=512, out_features=32128, bias=False)
)
```

In [51]:
```
new_model_arch_str = str(new_model)

with open("dwb-flan-t5-small-lora-finetune.model-architecture.txt", 'w', encoding='
    fhn.write(new_model_arch_str)
##endof:  with open ... fhn
```

@todo : get some Python version of `diff` going on here. I'm just using Cygwin/bash to see the LoRA additions.

Let's start by doing the single-dialogue summaries we used before.

Prompt (repeated for new kernel.

In [93]:
```
#def prompt_instruction_format(sample):
#    return f""" Instruction:
#      Use the Task below and the Input given to write the Response:
#
#      ### Task:
#      Summarize the Input
#
#      ### Input:
#      {sample['dialogue']}
#
#      ### Response:
#      {sample['summary']}
#      """
###endof:  prompt_instruction_format(sample)
```

Try one picked at random

In [94]:
```
#  Just one summarization to begin with, randomly picked ... but
#+ now with th possibility of a known seed, to allow visual
#+ comparison with after-training results.
#+ I'M NOT GOING TO USE THIS REPEATED SEED, I'm just going to
#+ use the datum at the first index to compare.

do_seed_for_repeatable = True

summarizer = pipeline('summarization',
                      model=new_model,
                      tokenizer=new_tokenizer)

if do_seed_for_repeatable:
    rand_seed_for_randrange = 137
    random.seed(rand_seed_for_randrange)
##endof:  if do_seed_for_repeatable

sample = dataset['test'][randrange(len(dataset["test"]))]
print(f"dialogue: \n{sample['dialogue']}\n---------------")

res = summarizer(sample["dialogue"])

print(f"dwb-flan-t5-small-lora-finetune summary:\n{res[0]['summary_text']}")
```

```
dialogue:
Jayden: But I don't need kids. Kids means over. At least for a woman
Brennan: Over what ?
Jayden: The end of normal life. Being pregnant, suffering because of this etc
Brennan: Hmm so I need to look for another mother to my kids then. Haha
Jayden: Being obligated to be with the. 24h. Men have only sex and they wait for kid
s while women suffer
Brennan: I don't agree...
Jayden: I wish I could do the same. Then probably i would say the same like u.
Brennan: Guys like me would be there through it all to reduce the suffering
Jayden: Physical suffering. No one can do anything with this. I wish I could just ha
ve sex and wait for a baby while having a normal life. Not getting fat, having the s
ame body, the same breast and not disgusting ... Not feeling sick, not having pain,
being able to do every day stuff even like walking...
Brennan: It's gonna happen eventually
Jayden: I was I'm a store, behind me there was a pregnant woman, she dropped some mo
ney and she couldn't even take them from the floor... I had to help her
Brennan: That's because she's about to give birth
Jayden: I hope that maybe soon they will be possible to have a child without being p
regnant. Yes! And she's suffering
Brennan: Any I'm sorry for feeding you with my bullshit
Jayden: While a man is doing his normal stuff. U mean the conversation?
Brennan: I hope you find a guy that can give you the sex you want and not get pregna
nt
Jayden: Would be awesome
Brennan: I'm gonna go to sleep now. Good night
Jayden: I said I don't want to have any children now! Maybe in the future when I hav
e a good job, I'm financially independent. Good night
---------------
dwb-flan-t5-small-lora-finetune summary:
I wish I could do the same. Then probably i would say the same like u. Brennan: Guys
like me would be there through it all to reduce the suffering Jayden: Physical suffe
ring. No one can do anything with this. I wishI could just have sex and wait for a b
aby while having a normal life. Not getting fat, having the same body, the same brea
st and not disgusting ... Not feeling sick, not having pain, being able to do every
day stuff even like walking... Brennan said It's gonna happen eventually Jaden: I wa
s I'm a store, behind me there was a pregnant woman, she dropped some money and she
couldn't even take them from the floor... I had to help her Brettan: That's because
she's about to give birth Jayden; I hope that maybe soon they will be possible to ha
ve a child without being pregnant. Yes
```

Now, a couple summarizations with comparison to ground truth

In [86]:
```
pred_test_list = []
ref_test_list = []

sample_num = 0

this_sample = dataset['test'][sample_num]

print(f"dialogue: \n{this_sample['dialogue']}\n---------------")

ground_summary = this_sample['summary']
res = summarizer(this_sample['dialogue'])
res_summary = res[0]['summary_text']

print(f"human-genratd summary:\n{ground_summary}")
print(f"dwb-flan-t5-small-lora-finetune summary:\n{res_summary}")

ref_test_list.append(ground_summary)
pred_test_list.append(res_summary)

# deprecated, blah blah blah
rouge = load_metric('rouge', trust_remote_code=True)

#  Yes, I have just one datum, but I'm setting things up to
#+ work well with a loop (meaning lists for pred and ref).
results_test_0 = rouge.compute(
                    predictions=pred_test_list,
                    references=ref_test_list,
                    use_aggregator=False
)

# >>> print(list(results_test.keys()))
# ['rouge1', 'rouge2', 'rougeL', 'rougeLsum']
```

```
Your max_length is set to 200, but your input_length is only 133. Since this is a su
mmarization task, where outputs shorter than the input are typically wanted, you mig
ht consider decreasing max_length manually, e.g. summarizer('...', max_length=66)
```

```
dialogue:
Hannah: Hey, do you have Betty's number?
Amanda: Lemme check
Hannah: <file_gif>
Amanda: Sorry, can't find it.
Amanda: Ask Larry
Amanda: He called her last time we were at the park together
Hannah: I don't know him well
Hannah: <file_gif>
Amanda: Don't be shy, he's very nice
Hannah: If you say so..
Hannah: I'd rather you texted him
Amanda: Just text him 🙂
Hannah: Urgh.. Alright
Hannah: Bye
Amanda: Bye bye
---------------
human-genratd summary:
Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry.
dwb-flan-t5-small-lora-finetune summary:
Amanda: Bye bye e. Alright Hannah: Hey, do you have Betty's number? Amanda: Lemme ch
eck Hannah: file_gif> Amanda: Sorry, can't find it. Amanda: Ask Larry Amanda: He cal
led her last time we were at the park together Hannah: I don'n't know him well Hanna
h: cfile_go-> Amanda: Don't be shy, he's very nice Hannah: If you say so.. Hannah: I
m'd rather you texted him Amanda: Just text him  Hannah: Urgh.. AlRIGHT Hannah: By e
Amanda: bye by e
```

In [87]:
```
print_rouge_scores(results_test_0, 0)
```

```
---------- ROUGE SCORES ----------
  --------- dialogue 1 ----------
ROUGE-1 results
[Score(
          precision=0.10465116279069768,
          recall=0.5625,
          fmeasure=0.17647058823529413)]
ROUGE-2 results
[Score(
          precision=0.023529411764705882,
          recall=0.1333333333333333,
          fmeasure=0.04)]
ROUGE-L results
[Score(
          precision=0.09302325581395349,
          recall=0.5,
          fmeasure=0.15686274509803924)]
ROUGE-Lsum results
[Score(
          precision=0.09302325581395349,
          recall=0.5,
          fmeasure=0.15686274509803924)]
```

In [95]:
```
sample_num = 224

this_sample = dataset['test'][sample_num]

print(f"dialogue: \n{this_sample['dialogue']}\n---------------")

ground_summary = this_sample['summary']
res = summarizer(this_sample['dialogue'])
res_summary = res[0]['summary_text']

print(f"human-genratd summary:\n{ground_summary}")
print(f"dwb-flan-t5-small-lora-finetune summary:\n{res_summary}")

ref_test_list = [ground_summary]
pred_test_list = [res_summary]

rouge = load_metric('rouge', trust_remote_code=True)

results_test_224 = rouge.compute(
                    predictions=pred_test_list,
                    references=ref_test_list,
                    use_aggregator=False
)

# >>> print(list(results_test.keys()))
# ['rouge1', 'rouge2', 'rougeL', 'rougeLsum']
```

```
dialogue:
Abigail: It's Sundaay.
Damien: So?..
Abigail: You know what that means.
Damien: Hmm no I don't x)
Abigail: Sunday means we go to church~.
Damien: Oh, yeah..
Abigail: Don't forget to put on a coat and tie.
Damien: A coat and tie?.. Why?
Abigail: To show respect to God and others.
Damien: Omg..I'm glad Sunday is only once a week.
Abigail: I hope God didn't hear that.
Damien: He'll forgive me 😇
Abigail: Just be ready on time please.
---------------
human-genratd summary:
Abigail and Damien are going to church on Sunday. Damien has to put on a coat and ti
e.
dwb-flan-t5-small-lora-finetune summary:
Abigail: Sunday means we go to church. Damien: Oh, yeah.. Abigaill: Don't forget to
put on a coat and tie. Damen: A coat and tied?.. Why? Abigaile: To show respect to G
od and others. Damian: Omg..I'm glad Sunday is only once a week. Abguril: I hope God
didn't hear that. Damier: He'll forgive me  Abigails: Just be ready on time please.
```

In [96]:
```
print_rouge_scores(results_test_224, 224)
```

```
---------- ROUGE SCORES ----------
  --------- dialogue 225 ----------
ROUGE-1 results
[Score(
          precision=0.21212121212121213,
          recall=0.7777777777777778,
          fmeasure=0.3333333333333333)]
ROUGE-2 results
[Score(
          precision=0.1076923076923077,
          recall=0.4117647058823529,
          fmeasure=0.1707317073170732)]
ROUGE-L results
[Score(
          precision=0.16666666666666666,
          recall=0.6111111111111112,
          fmeasure=0.2619047619047619)]
ROUGE-Lsum results
[Score(
          precision=0.16666666666666666,
          recall=0.6111111111111112,
          fmeasure=0.2619047619047619)]
```

In [ ]:

In [ ]:

In [ ]:

Evaluation on the Test Set and Comparison to Baseline

Verbosity stuff - get rid of the nice advice

In [ ]:
```
# # Don't need this again
```

In [90]:
```
log_verbosity_is_critical = \
  logging.get_verbosity() == logging.CRITICAL # alias FATAL, 50
log_verbosity_is_error = \
  logging.get_verbosity() == logging.ERROR # 40
log_verbosity_is_warn = \
  logging.get_verbosity() == logging.WARNING # alias WARN, 30
log_verbosity_is_info = \
  logging.get_verbosity() == logging.INFO # 20
log_verbosity_is_debug = \
  logging.get_verbosity() == logging.DEBUG # 10
```

```
print( "The statement, 'logging verbosity is CRITICAL' " + \
      f"is {log_verbosity_is_critical}")
print( "The statement, 'logging verbosity is    ERROR' " + \
      f"is {log_verbosity_is_error}")
print( "The statement, 'logging verbosity is  WARNING' " + \
      f"is {log_verbosity_is_warn}")
print( "The statement, 'logging verbosity is     INFO' " + \
      f"is {log_verbosity_is_info}")
print( "The statement, 'logging verbosity is    DEBUG' " + \
      f"is {log_verbosity_is_debug}")

print()

init_log_verbosity = logging.get_verbosity()
print(f"The value of logging.get_verbosity() is: {init_log_verbosity}")

print()

init_t_n_a_w = os.environ.get('TRANSFORMERS_NO_ADVISORY_WARNINGS')
print(f"TRANSFORMERS_NO_ADIVSORY_WARNINGS: {init_t_n_a_w}")
```

```
The statement, 'logging verbosity is CRITICAL' is False
The statement, 'logging verbosity is    ERROR' is False
The statement, 'logging verbosity is  WARNING' is True
The statement, 'logging verbosity is     INFO' is False
The statement, 'logging verbosity is    DEBUG' is False

The value of logging.get_verbosity() is: 30

TRANSFORMERS_NO_ADIVSORY_WARNINGS: None
```

Here's the actual evaluation

In [62]:
```
# # Don't need this again
```

!!! NOTE !!! I'm going to use `tat` (with an underscore or undescores before, after, or surrounding the variable names) to indicate 'testing-after-training'.

I guess I could have used `inference`, but I didn't.

!!! another NOTE You'd better **make dang sure you want the lots of output** before you set this next boolean to `True`

In [63]:
```
do_have_lotta_output_from_all_dialogs_summaries = False
```

Are you sure about the value of that last boolean?

There could be megabytes (maybe gigabytes) worth of text output if you've changed it to `True`.

Evaluation on the Test Set and Comparison to Baseline

In [91]:
```
logging.set_verbosity_error()

tat_summarizer = pipeline('summarization',
                          model=new_model,
                          tokenizer=new_tokenizer)

#*p*#tat_sample_dialog_list = []
prediction_tat_list = []
reference_tat_list = []

tat_tic = timeit.default_timer()

for sample_num in range(len(dataset['test'])):
    this_sample = dataset['test'][sample_num]

    if do_have_lotta_output_from_all_dialogs_summaries:
        print("="*75)
        print(f"dialogue: \n{this_sample['dialogue']}\n---------------")
    ##endof:  if do_have_lotta_output_from_all_dialogs_summaries

    ground_tat_summary = this_sample['summary']
    res_tat = summarizer(this_sample['dialogue'])
    res_tat_summary = res_tat[0]['summary_text']

    if do_have_lotta_output_from_all_dialogs_summaries:
        print("-"*70)
        print(f"human-genratd summary:\n{ground_tat_summary}")
        print("-"*70)
```

```
        print( "dwb-flan-t5-small-lora-finetune summary:" + \
              f"\n{res_tat_summary}")
        print("-"*70)
    ##endof:  if do_have_lotta_output_from_all_dialogs_summaries

    #*p*#    tat_sample_dialog_list.append(this_sample)
    reference_tat_list.append(ground_tat_summary)
    prediction_tat_list.append(res_tat_summary)
##endof:  for sample_num in range(len(dataset['test']))

tat_toc = timeit.default_timer()

tat_duration = tat_toc - tat_tic

print( "Getting things ready for scoring (after training)")
print(f"took {tat_toc - tat_tic:0.4f} seconds.")

tat_time_str = format_timespan(tat_duration)

print(f"which equates to {tat_time_str}")


rouge = load_metric('rouge', trust_remote_code=True)

results_tat = rouge.compute(
                  predictions=prediction_tat_list,
                  references=reference_tat_list,
                  use_aggregator=True
)

# >>> print(list(results_tat.keys()))
# ['rouge1', 'rouge2', 'rougeL', 'rougeLsum']

#*p*# objects_to_pickle.append(tat_sample_dialog_list)
#*p*# objects_to_pickle.append(prediction_tat_list)
#*p*# objects_to_pickle.append(reference_tat_list)
#*p*# objects_to_pickle.append(results_tat)
```

```
Getting things ready for scoring (after training)
took 3941.8285 seconds.
which equates to 1 hour, 5 minutes and 41.83 seconds
```

In [65]:
```
print_rouge_scores(results_tat, "TEST AFTER TRAINING")
```

```
---------- ROUGE SCORES ----------
  --------- TEST AFTER TRAINING ----------
ROUGE-1 results
AggregateScore(
     low=Score(
          precision=0.18465960106995058,
          recall=0.5289884514472354,
          fmeasure=0.25686215590159345),
     mid=Score(
          precision=0.19191206582001252,
          recall=0.5419927875442789,
          fmeasure=0.26514311109911903),
     high=Score(
          precision=0.19892074709381968,
          recall=0.5560562002147722,
          fmeasure=0.273181138437335)
)
ROUGE-2 results
AggregateScore(
     low=Score(
          precision=0.05269906298279127,
          recall=0.15575094190620362,
          fmeasure=0.07409348910518994),
     mid=Score(
          precision=0.05716364568273007,
          recall=0.16594455254812504,
          fmeasure=0.0797410330836727),
     high=Score(
          precision=0.06147635605696184,
          recall=0.1761782280013389,
          fmeasure=0.08508543913464939)
)
ROUGE-L results
AggregateScore(
     low=Score(
          precision=0.1358391419777274,
          recall=0.38856823315589245,
          fmeasure=0.18868402958397204),
     mid=Score(
          precision=0.1413916125834373,
          recall=0.39950997023341217,
          fmeasure=0.19515605909360623),
     high=Score(
          precision=0.14730026614718988,
          recall=0.4117337256634965,
          fmeasure=0.20223694130284198)
)
ROUGE-Lsum results
AggregateScore(
     low=Score(
          precision=0.13581244201168188,
          recall=0.38871271829792664,
          fmeasure=0.1886832040731757),
     mid=Score(
          precision=0.14133285520379574,
```

```
                    recall=0.3998294223955289,
                    fmeasure=0.1950362847385175),
               high=Score(
                    precision=0.147008914964092,
                    recall=0.4109702578189206,
                    fmeasure=0.20206485129716942)
          )
```

In [66]:
```
##  Haven't tried this, because the logging seemed easier,
##+ and the logging worked
# os.environ("TRANSFORMERS_NO_ADVISORY_WARNINGS") = init_t_n_a_w

logging.set_verbosity(init_log_verbosity)
```

In [ ]:
```

```

Pickle things to pickle save

In [ ]:
```
with open(pickle_filename, 'wb') as pfh:
    pickle.dump(objects_to_pickle , pfh)
##endof:  with open ... as pfh # (pickle file handle)
```

Notes Looking Forward to LoRA on RWKV

Hugging Face Community, seems to have a good portion of their models

https://huggingface.co/RWKV

https://web.archive.org/web/20240530232509/https://huggingface.co/RWKV

GitHub has even more versions/models, including the `v4-neo` that I think will be important (the LoRA project)

https://github.com/BlinkDL/RWKV-LM/tree/main

https://web.archive.org/web/20240530232637/https://github.com/BlinkDL/RWKV-LM/tree/main

The main RWKV website (?!)

https://www.rwkv.com/

https://web.archive.org/web/20240529120904/https://www.rwkv.com/

GOOD STUFF. A project doing LoRA with RWKV

https://github.com/Blealtan/RWKV-LM-LoRA/

https://web.archive.org/web/20240530232823/https://github.com/Blealtan/RWKV-LM-LoRA

The official blog, I guess, with some good coding examples

https://huggingface.co/blog/rwkv

https://web.archive.org/web/20240530233025/https://huggingface.co/blog/rwkv

It includes something that's similar to what I'm doing here in the `First_Full_LoRA_Trial_with_Transformer_Again.ipynb` tutorial, etc.

```
from transformers import AutoTokenizer, AutoModelForCausalLM

model_id = "RWKV/rwkv-raven-1b5"

model = AutoModelForCausalLM.from_pretrained(model_id).to(0)
tokenizer = AutoTokenizer.from_pretrained(model_id)
```

The `AutoModelForCausalLM` is the same as the tutorial I'm following, but I don't know what the `.to(0)` is for.

Really quickly, also looking at

https://huggingface.co/RWKV/rwkv-4-world-7b

https://web.archive.org/web/20240530234438/https://huggingface.co/RWKV/rwkv-4-world-7b

I see an example for CPU.

```
model = AutoModelForCausalLM.from_pretrained(
              "RWKV/rwkv-4-world-7b",
              trust_remote_code=True
).to(torch.float32)

tokenizer = AutoTokenizer.from_pretrained(
              "RWKV/rwkv-4-world-7b",
              trust_remote_code=True)
```

(Old version? Unofficial, it seems)

https://huggingface.co/docs/transformers/en/model_doc/rwkv

https://web.archive.org/web/20240530232341/https://huggingface.co/docs/transformers/en/mo

In [ ]: