First Full LoRA Trial with Transformer

What We Are Doing

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

Starting

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 [1]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y%m%dT%H%M%S%Z00\") -replace '[.][0-9]*_', '_'
```

```
1717845907_20240608T112507-0600
```

Output was:

`timestamp`

Installation - minimize it unless you need `!pip install` commands

(Feel free to drop down to the TL;DR section.)

Detailed stuff - minimize it

Conda environment file - not recommended, minimize

My ~~`environment.yml` file for a cpu (specifically, one run on Windows)~~ `environment-cpu.yml` will have its contents listed below. It should have everything needed for an install anywhere. The directory should have a `full_environment-cpu.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 `environment.yml` file (not necessarily the same as the `environment-cpu.yml` 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.

Note that the one below was written for a CPU, and it was run on Windows. Since then, I also have an `environment.yml` file written for CoLab (Linux(R)). There are files with `environment-cpu.yml` and `full_environment-cpu.yml` as well as with `environment-colab.yml` and `full_environment-colab.yml`. Make sure the one you have matches your computer's setup.

In [2]:
```
do_want_to_read_realtime = True

if do_want_to_read_realtime:
    with open("environment-cpu.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
```

For the CPU, I got the following.

```
# @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
```

Possible conda then pip install - recommended

If packages aren't installed - a situation that happens every time with CoLab and that will happen the first time on my CPU-with-Windows-in-the-corner-compy-with-three-screens - this is the suggested way to go.

Local Machine

1-local) From a local machine (First time only)

(a) (First time only) Start from a shell. Create a conda environment; I do it from `Conda Prompt (miniconda3)`. conda create -n rwkv-lora-first python=3.10.14 conda activate rwkv-lora-first pip install --upgrade pip==24.0 # note that you should use the `--upgrade` whether upgrading or downgrading pip pip install jupyter==1.0.0 jupyter notebook First_Full_LoRA_Trial_with_Transformer_Again.ipynb

(The notebook, and the conda environment for that matter, might have different names.) **(b)** (First time only) Installations.

(i) Go to the `!pip install` parts (Install TL;DR), making sure that you set the `do_pip_installs` boolean to `True`.

(ii) Do the installs.

(iii) [Optional] Run the `!pip freeze` and copy/paste the results to an info spot for local machine.

2-local) (First and subsequent times) Start from a shell. Activate the conda environment and launch the Jupyter notebook.

```
conda activate rwkv-lora-first
jupyter notebook First_Full_LoRA_Trial_with_Transformer_Again.ipynb
```

(The notebook, and the conda environment for that matter, might have different names.)

3-local) (Subsequent times) Do not re-run the `!pip install` commands unless something has gone wrong. Make sure that you set the `do_pip_installs` boolean to `False`.

CoLab

1-CoLab) (Do all steps every time.)

(a) go to https://colab.research.google.com/

(b) Choose the GitHub Option.

(i) In the `Enter a GitHub URL or search by organization or user` text box, put in `bballdave025`

(ii) In the `Repository` drop-down menu, choose `bballdave025/rwkv-lora`

(iii) In the `Branch` drop-down menu, choose either `fix-nb-so-all-executes-everywhere`. It's possible that `main` might have some work being done and thus might not execute.

(iv) From the choices (links) below, choose `first_full_LoRA trial_w_Transformer/First_Full_Lora_with_Transformer_Again_4Colab.ipynb`

(c) Go to the `!pip install` parts (Install TL;DR), making sure that you set the `do_pip_installs` boolean to `True`.

2-CoLab) [Optional] Run the `!pip freeze` and copy/paste the results to an info spot for CoLab.

Note - minimize and don't use

If you are in this Jupyter Notebook but aren't in a `conda` environment ... and if you know enough to realize that and to know what the following commands do, you can uncomment the commands below to get your `conda` environment set up.

In [3]:
```
## not going to do complicated subprocess stuff here. Sorry.
```

Install TL;DR

In [4]:
```
##  If the installs have been completed - either through conda
##+ or through having previously run this notebook, this boolean
##+ should have a `False` value

do_pip_installs = True
```

Once you have things ready and a jupyter notebook running, you can do the `!pip install` commands that follow, depending on whether you are in CoLab or on a local machine, and whether or not it's the first time.

In [ ]:
```
if do_pip_installs:
    !pip install --upgrade pip==24.0
##endof:  if do_pip_installs
```

In [5]:
```
if do_pip_installs:
    !pip install accelerate bitsandbytes evaluate datasets huggingface-hub
    !pip install humanfriendly nltk py-cpuinfo pylspci rouge-score
    !pip install tensorflow-cpu torch transformers trl
##endof:  if do_pip_installs
```

Trying this next one on its own, since it might fail when not on Windows. **Update:** This installs fine on CoLab.

In [6]:
```
if do_pip_installs:
    !pip install wmi
##endof:  if do_pip_installs
```

And now, for the installs from GitHub repos

In [7]:
```
if do_pip_installs:
    !pip install git+https://github.com/huggingface/peft.git@e7b75070c72a88f0f7926cc6872858a2c5f0090d
##endof:  if do_pip_installs
```

In [8]:
```
if do_pip_installs:
    !pip install git+https://github.com/nexplorer-3e/qwqfetch.git@f72d222e2fff5ffea9f4e4b3a203e4c4d9e8cf00
##endof:  if do_pip_installs
```

Specifics for CoLab Environment - minimize it

In [9]:
```
# # Don't need this again
#!date +'%s_%Y%m%dT%H%M%S%z'
```

Output was:

`timestamp`

In [10]:
```
#!pip freeze
```

For CoLab, I got the following from `!pip freeze`

```
absl-py==1.4.0
accelerate==0.30.1
aiohttp==3.9.5
aiosignal==1.3.1
alabaster==0.7.16
albumentations==1.3.1
altair==4.2.2
annotated-types==0.7.0
anyio==3.7.1
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
array_record==0.5.1
arviz==0.15.1
astropy==5.3.4
```

```
astunparse==1.6.3
async-timeout==4.0.3
atpublic==4.1.0
attrs==23.2.0
audioread==3.0.1
autograd==1.6.2
Babel==2.15.0
backcall==0.2.0
beautifulsoup4==4.12.3
bidict==0.23.1
bigframes==1.8.0
bitsandbytes==0.43.1
bleach==6.1.0
blinker==1.4
blis==0.7.11
blosc2==2.0.0
bokeh==3.3.4
bqplot==0.12.43
branca==0.7.2
build==1.2.1
CacheControl==0.14.0
cached-property==1.5.2
cachetools==5.3.3
catalogue==2.0.10
certifi==2024.6.2
cffi==1.16.0
chardet==5.2.0
charset-normalizer==3.3.2
chex==0.1.86
click==8.1.7
click-plugins==1.1.1
cligj==0.7.2
cloudpathlib==0.16.0
cloudpickle==2.2.1
cmake==3.27.9
cmdstanpy==1.2.3
colorcet==3.1.0
colorlover==0.3.0
colour==0.1.5
```

```
community==1.0.0b1
confection==0.1.5
cons==0.4.6
contextlib2==21.6.0
contourpy==1.2.1
cryptography==42.0.7
cuda-python==12.2.1
cudf-cu12 @ \\
https://pypi.nvidia.com/cudf-cu12/\\
cudf_cu12-24.4.1-cp310-cp310-\\
manylinux_2_28_x86_64.whl#\\
sha256=57366e7ef09dc63e0b389aff20d\\
f6c37d91e2790065861ee31a4720149f5b694
cufflinks==0.17.3
cupy-cuda12x==12.2.0
cvxopt==1.3.2
cvxpy==1.3.4
cycler==0.12.1
cymem==2.0.8
Cython==3.0.10
dask==2023.8.1
datascience==0.17.6
datasets==2.19.2
db-dtypes==1.2.0
dbus-python==1.2.18
debugpy==1.6.6
decorator==4.4.2
defusedxml==0.7.1
dill==0.3.8
distributed==2023.8.1
distro==1.7.0
dlib==19.24.4
dm-tree==0.1.8
docstring_parser==0.16
docutils==0.18.1
dopamine_rl==4.0.9
duckdb==0.10.3
earthengine-api==0.1.405
easydict==1.13
```

```
ecos==2.0.13
editdistance==0.6.2
eerepr==0.0.4
en-core-web-sm @ \\
https://github.com/explosion/spacy-models/\\
releases/download/en_core_web_sm-3.7.1/\\
en_core_web_sm-3.7.1-py3-none-any.whl#\\
sha256=86cc141f63942d4b2c5fcee06630fd6f904\\
788d2f0ab005cce45aadb8fb73889
entrypoints==0.4
et-xmlfile==1.1.0
etils==1.7.0
etuples==0.3.9
evaluate==0.4.2
exceptiongroup==1.2.1
fastai==2.7.15
fastcore==1.5.43
fastdownload==0.0.7
fastjsonschema==2.19.1
fastprogress==1.0.3
fastrlock==0.8.2
filelock==3.14.0
fiona==1.9.6
firebase-admin==5.3.0
Flask==2.2.5
flatbuffers==24.3.25
flax==0.8.4
folium==0.14.0
fonttools==4.53.0
frozendict==2.4.4
frozenlist==1.4.1
fsspec==2023.6.0
future==0.18.3
gast==0.5.4
gcsfs==2023.6.0
GDAL==3.6.4
gdown==5.1.0
geemap==0.32.1
gensim==4.3.2
```

```
geocoder==1.38.1
geographiclib==2.0
geopandas==0.13.2
geopy==2.3.0
gin-config==0.5.0
glob2==0.7
google==2.0.3
google-ai-generativelanguage==0.6.4
google-api-core==2.11.1
google-api-python-client==2.84.0
google-auth==2.27.0
google-auth-httplib2==0.1.1
google-auth-oauthlib==1.2.0
google-cloud-aiplatform==1.52.0
google-cloud-bigquery==3.21.0
google-cloud-bigquery-connection==1.12.1
google-cloud-bigquery-storage==2.25.0
google-cloud-core==2.3.3
google-cloud-datastore==2.15.2
google-cloud-firestore==2.11.1
google-cloud-functions==1.13.3
google-cloud-iam==2.15.0
google-cloud-language==2.13.3
google-cloud-resource-manager==1.12.3
google-cloud-storage==2.8.0
google-cloud-translate==3.11.3
google-colab @ \\
file:///colabtools/dist/google-colab-1.0.0.tar.gz#\\
sha256=4eddb762b7958cca5d83aaa7251a8bfb129afc8608a\\
2823c2d0a6770a5c27bc4
google-crc32c==1.5.0
google-generativeai==0.5.4
google-pasta==0.2.0
google-resumable-media==2.7.0
googleapis-common-protos==1.63.1
googledrivedownloader==0.4
graphviz==0.20.3
greenlet==3.0.3
grpc-google-iam-v1==0.13.0
```

```
grpcio==1.64.1
grpcio-status==1.48.2
gspread==6.0.2
gspread-dataframe==3.3.1
gym==0.25.2
gym-notices==0.0.8
h5netcdf==1.3.0
h5py==3.9.0
holidays==0.50
holoviews==1.17.1
html5lib==1.1
httpimport==1.3.1
httplib2==0.22.0
huggingface-hub==0.23.2
humanfriendly==10.0
humanize==4.7.0
hyperopt==0.2.7
ibis-framework==8.0.0
idna==3.7
imageio==2.31.6
imageio-ffmpeg==0.5.1
imagesize==1.4.1
imbalanced-learn==0.10.1
imgaug==0.4.0
immutabledict==4.2.0
importlib_metadata==7.1.0
importlib_resources==6.4.0
imutils==0.5.4
inflect==7.0.0
iniconfig==2.0.0
intel-openmp==2023.2.4
ipyevents==2.0.2
ipyfilechooser==0.6.0
ipykernel==5.5.6
ipyleaflet==0.18.2
ipython==7.34.0
ipython-genutils==0.2.0
ipython-sql==0.5.0
ipytree==0.2.2
```

```
ipywidgets==7.7.1
itsdangerous==2.2.0
jax==0.4.26
jaxlib @ \\
https://storage.googleapis.com/jax-releases/\\
cuda12/jaxlib-0.4.26+cuda12.cudnn89-cp310-cp310\\
-manylinux2014_x86_64.whl#\\
sha256=813cf1fe3e7ca4dbf5327d6e7b4fc8521e92d8bb\\
a073ee645ae0d5d036a25750
jeepney==0.7.1
jellyfish==1.0.4
jieba==0.42.1
Jinja2==3.1.4
joblib==1.4.2
jsonpickle==3.0.4
jsonschema==4.19.2
jsonschema-specifications==2023.12.1
jupyter-client==6.1.12
jupyter-console==6.1.0
jupyter-server==1.24.0
jupyter_core==5.7.2
jupyterlab_pygments==0.3.0
jupyterlab_widgets==3.0.11
kaggle==1.6.14
kagglehub==0.2.5
keras==2.15.0
keyring==23.5.0
kiwisolver==1.4.5
langcodes==3.4.0
language_data==1.2.0
launchpadlib==1.10.16
lazr.restfulclient==0.14.4
lazr.uri==1.0.6
lazy_loader==0.4
libclang==18.1.1
librosa==0.10.2.post1
lightgbm==4.1.0
linkify-it-py==2.0.3
llvmlite==0.41.1
```

```
locket==1.0.0
logical-unification==0.4.6
lxml==4.9.4
malloy==2023.1067
marisa-trie==1.1.1
Markdown==3.6
markdown-it-py==3.0.0
MarkupSafe==2.1.5
matplotlib==3.7.1
matplotlib-inline==0.1.7
matplotlib-venn==0.11.10
mdit-py-plugins==0.4.1
mdurl==0.1.2
miniKanren==1.0.3
missingno==0.5.2
mistune==0.8.4
mizani==0.9.3
mkl==2023.2.0
ml-dtypes==0.2.0
mlxtend==0.22.0
more-itertools==10.1.0
moviepy==1.0.3
mpmath==1.3.0
msgpack==1.0.8
multidict==6.0.5
multipledispatch==1.0.0
multiprocess==0.70.16
multitasking==0.0.11
murmurhash==1.0.10
music21==9.1.0
natsort==8.4.0
nbclassic==1.1.0
nbclient==0.10.0
nbconvert==6.5.4
nbformat==5.10.4
nest-asyncio==1.6.0
networkx==3.3
nibabel==4.0.2
nltk==3.8.1
```

```
notebook==6.5.5
notebook_shim==0.2.4
numba==0.58.1
numexpr==2.10.0
numpy==1.25.2
nvidia-cublas-cu12==12.1.3.1
nvidia-cuda-cupti-cu12==12.1.105
nvidia-cuda-nvrtc-cu12==12.1.105
nvidia-cuda-runtime-cu12==12.1.105
nvidia-cudnn-cu12==8.9.2.26
nvidia-cufft-cu12==11.0.2.54
nvidia-curand-cu12==10.3.2.106
nvidia-cusolver-cu12==11.4.5.107
nvidia-cusparse-cu12==12.1.0.106
nvidia-nccl-cu12==2.20.5
nvidia-nvjitlink-cu12==12.5.40
nvidia-nvtx-cu12==12.1.105
nvtx==0.2.10
oauth2client==4.1.3
oauthlib==3.2.2
opencv-contrib-python==4.8.0.76
opencv-python==4.8.0.76
opencv-python-headless==4.10.0.82
openpyxl==3.1.3
opt-einsum==3.3.0
optax==0.2.2
orbax-checkpoint==0.4.4
osqp==0.6.2.post8
packaging==24.0
pandas==2.0.3
pandas-datareader==0.10.0
pandas-gbq==0.19.2
pandas-stubs==2.0.3.230814
pandocfilters==1.5.1
panel==1.3.8
param==2.1.0
parso==0.8.4
parsy==2.1
partd==1.4.2
```

```
pathlib==1.0.1
patsy==0.5.6
peewee==3.17.5
peft @ \\
git+https://github.com/huggingface/peft.git@\\
03798a9143c90d796a0bee8f43863668d084381f
pexpect==4.9.0
pickleshare==0.7.5
Pillow==9.4.0
pip-tools==6.13.0
platformdirs==4.2.2
plotly==5.15.0
plotnine==0.12.4
pluggy==1.5.0
polars==0.20.2
pooch==1.8.1
portpicker==1.5.2
prefetch-generator==1.0.3
preshed==3.0.9
prettytable==3.10.0
proglog==0.1.10
progressbar2==4.2.0
prometheus_client==0.20.0
promise==2.3
prompt_toolkit==3.0.45
prophet==1.1.5
proto-plus==1.23.0
protobuf==3.20.3
psutil==5.9.5
psycopg2==2.9.9
ptyprocess==0.7.0
py-cpuinfo==9.0.0
py4j==0.10.9.7
pyarrow==14.0.2
pyarrow-hotfix==0.6
pyasn1==0.6.0
pyasn1_modules==0.4.0
pycocotools==2.0.7
pycparser==2.22
```

```
pydantic==2.7.3
pydantic_core==2.18.4
pydata-google-auth==1.8.2
pydot==1.4.2
pydot-ng==2.0.0
pydotplus==2.0.2
PyDrive==1.3.1
PyDrive2==1.6.3
pyerfa==2.0.1.4
pygame==2.5.2
Pygments==2.16.1
PyGObject==3.42.1
PyJWT==2.3.0
pylspci==0.4.3
pymc==5.10.4
pymystem3==0.2.0
pynvjitlink-cu12==0.2.3
PyOpenGL==3.1.7
pyOpenSSL==24.1.0
pyparsing==3.1.2
pyperclip==1.8.2
pyproj==3.6.1
pyproject_hooks==1.1.0
pyshp==2.3.1
PySocks==1.7.1
pytensor==2.18.6
pytest==7.4.4
python-apt @ file:///backend-container/containers/\\
python_apt-0.0.0-cp310-cp310-linux_x86_64.whl#\\
sha256=b209c7165d6061963abe611492f8c91c3bcef4b\\
7a6600f966bab58900c63fefa
python-box==7.1.1
python-dateutil==2.8.2
python-louvain==0.16
python-slugify==8.0.4
python-utils==3.8.2
pytz==2023.4
pyviz_comms==3.0.2
PyWavelets==1.6.0
```

```
PyYAML==6.0.1
pyzmq==24.0.1
qdldl==0.1.7.post2
qudida==0.0.4
qwqfetch @ \\
git+https://github.com/nexplorer-3e/\\
qwqfetch.git@\\
f72d222e2fff5ffea9f4e4b3a203e4c4d9e8cf00
ratelim==0.1.6
referencing==0.35.1
regex==2024.5.15
requests==2.32.3
requests-oauthlib==1.3.1
requirements-parser==0.9.0
rich==13.7.1
rmm-cu12==24.4.0
rouge-score==0.1.2
rpds-py==0.18.1
rpy2==3.4.2
rsa==4.9
safetensors==0.4.3
scikit-image==0.19.3
scikit-learn==1.2.2
scipy==1.11.4
scooby==0.10.0
scs==3.2.4.post2
seaborn==0.13.1
SecretStorage==3.3.1
Send2Trash==1.8.3
sentencepiece==0.1.99
shapely==2.0.4
shtab==1.7.1
simple_parsing==0.1.5
six==1.16.0
sklearn-pandas==2.2.0
smart-open==6.4.0
sniffio==1.3.1
snowballstemmer==2.2.0
sortedcontainers==2.4.0
```

```
soundfile==0.12.1
soupsieve==2.5
soxr==0.3.7
spacy==3.7.4
spacy-legacy==3.0.12
spacy-loggers==1.0.5
Sphinx==5.0.2
sphinxcontrib-applehelp==1.0.8
sphinxcontrib-devhelp==1.0.6
sphinxcontrib-htmlhelp==2.0.5
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==1.0.7
sphinxcontrib-serializinghtml==1.1.10
SQLAlchemy==2.0.30
sqlglot==20.11.0
sqlparse==0.5.0
srsly==2.4.8
stanio==0.5.0
statsmodels==0.14.2
StrEnum==0.4.15
sympy==1.12.1
tables==3.8.0
tabulate==0.9.0
tbb==2021.12.0
tblib==3.0.0
tenacity==8.3.0
tensorboard==2.15.2
tensorboard-data-server==0.7.2
tensorflow @ \\
https://storage.googleapis.com/\\
colab-tf-builds-public-09h6ksrfwbb9g9xv/\\
tensorflow-2.15.0-cp310-cp310-\\
manylinux_2_17_x86_64.manylinux2014_x86_64.whl#\\
sha256=a2ec79931350b378c1ef300ca836b52a55751acb\\
71a433582508a07f0de57c42
tensorflow-datasets==4.9.5
tensorflow-estimator==2.15.0
tensorflow-gcs-config==2.15.0
tensorflow-hub==0.16.1
```

```
tensorflow-io-gcs-filesystem==0.37.0
tensorflow-metadata==1.15.0
tensorflow-probability==0.23.0
tensorstore==0.1.45
termcolor==2.4.0
terminado==0.18.1
text-unidecode==1.3
textblob==0.17.1
tf-slim==1.1.0
tf_keras==2.15.1
thinc==8.2.3
threadpoolctl==3.5.0
tifffile==2024.5.22
tinycss2==1.3.0
tokenizers==0.19.1
toml==0.10.2
tomli==2.0.1
toolz==0.12.1
torch @ \\
https://download.pytorch.org/whl/cu121/\\
torch-2.3.0%2Bcu121-cp310-cp310-linux_x86_64.whl#\\
sha256=0a12aa9aa6bc442dff8823ac8b48d991fd0771562e\\
aa38593f9c8196d65f7007
torchaudio @ \\
https://download.pytorch.org/whl/cu121/\\
torchaudio-2.3.0%2Bcu121-cp310-cp310-linux_x86_64.whl#\\
sha256=38b49393f8c322dcaa29d19e5acbf5a0b1978cf1b719445\\
ab670f1fb486e3aa6
torchsummary==1.5.1
torchtext==0.18.0
torchvision @ \\
https://download.pytorch.org/whl/cu121/\\
torchvision-0.18.0%2Bcu121-cp310-cp310-linux_x86_64.whl#\\
sha256=13e1b48dc5ce41ccb8100ab3dd26fdf31d8f1e904ecf2865a\\
c524493013d0df5
tornado==6.3.3
tqdm==4.66.4
traitlets==5.7.1
traittypes==0.2.1
```

```
transformers==4.41.2
triton==2.3.0
trl==0.9.4
tweepy==4.14.0
typer==0.9.4
types-pytz==2024.1.0.20240417
types-setuptools==70.0.0.20240524
typing_extensions==4.12.1
tyro==0.8.4
tzdata==2024.1
tzlocal==5.2
uc-micro-py==1.0.3
uritemplate==4.1.1
urllib3==2.0.7
vega-datasets==0.9.0
wadllib==1.3.6
wasabi==1.1.3
wcwidth==0.2.13
weasel==0.3.4
webcolors==1.13
webencodings==0.5.1
websocket-client==1.8.0
Werkzeug==3.0.3
widgetsnbextension==3.6.6
WMI==1.4.9
wordcloud==1.9.3
wrapt==1.14.1
xarray==2023.7.0
xarray-einstats==0.7.0
xgboost==2.0.3
xlrd==2.0.1
xxhash==3.4.1
xyzservices==2024.4.0
yarl==1.9.4
yellowbrick==1.5
yfinance==0.2.40
zict==3.0.0
zipp==3.19.1
```

Specifics for Windows CPU Environment - minimize it

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

Output was:

`timestamp`

In [ ]:
```
!pip freeze
```

For Windows CPU, I got the following from `!pip freeze`

```
Output from pip freeze
```

Imports

On a local machine (not on CoLab) you shouldn't need to know about the files in the current working directory, other than as information that might be useful in understanding where we're loading datasets or where the modules I've made live.

Modules: `system_info_as_script.py` and `dwb_rouge_scores.py`

Dataset files: `samsum-train.json` , `samsum-validation.json` , and `samsum-test.json` .

In [11]:
```
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, \
                AutoPeftModelForSeq2SeqLM, \
                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
import dwb_rouge_scores
```

In [12]:
```
# #  Don't need for local machine
#os.getcwd()
```

Load the training and test dataset along with the LLM and 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.)~~

Update: It seems that validation-set use with the trainer wasn't part of the example.

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

Update: It (running the validation set) worked fine, though its loss (the loss on the validation set) is lower than the training set's loss.

In [13]:
```
#  Need to install  datasets  (i.e. the `datasets` module/package)
#+ from `pip`, not `conda`. I'll do all from `pip`.
#+
#+ 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()
print("Getting the 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
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
Loading the original model, google/flan-t5-small
took 0.8088 seconds.
which equates to 0.81 seconds

Getting the original tokenizer
took 0.2681 seconds.
which equates to 0.27 seconds
```

I wonder if those lines,

```
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

will be the same for RWKV.

Notes from trying to get rid of weird output - minimize

I've thought about changing the line

`model = AutoModelForSeq2SeqLM.from_pretrained(model_name)`

to match the `peft` configuration, i.e.

> ```
> peft_config = LoraConfig( lora_alpha=16,
>                           lora_dropout=0.1,
>                           r=64,
>                           bias='none',
>                           task_type='CAUSAL_LM',
> )
> ```

I've thought about using

`model = AutoModelForCausalLM.from_pretrained(model_name)`

but every documentation I've consulted uses the `Seq2SeqLM`. e.g.

```
doc1 = "https://web.archive.org/web/20240506213344/" + \\
       "https://huggingface.co/docs/transformers/en/" + \\
       "model_doc/flan-t5"
```

Also, there is the info from

```
doc2="https://huggingface.co/transformers/v3.0.2/model_doc/t5.html"
```

> T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which **each task is converted into a text-to-text format.**

Something similar is in the paper abstract for

https://arxiv.org/pdf/1910.10683.pdf

Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified **Text-to-Text** Transformer". online. arXiv:cs.LG.1910.10683v4. 19 Sep 2023. retrieved 06 June 2024

which is cited in `doc2`

> In this paper, we explore the landscape of transfer learning techniques for NLP by introducing a unified framework that converts all text-based language problems into a **text-to-text format**.

(All emphasis is mine, DWB.)

Google Results

As of today (2024-06-06), a Google search for

`"AutoModelForCausalLM from_pretrained google flan-t5-small"`

(with quotes) returns

> Your search - "AutoModelForCausalLM from_pretrained google flan-t5-small" - did not match any documents.
>
> Suggestions:
>
> - Make sure all words are spelled correctly.
> - Try different keywords.
> - Try more general keywords.

whereas a Google search (again with quotes) for

`"AutoModelForSeq2SeqLM from_pretrained google flan-t5-small"`

returns

> About 119 results (0.22 seconds)

Trying the experiment

With all that, I tried the line anyway. Using just the important lines

`IN:`

```
model_name = "google/flan-t5-small"
```

…

```
model = AutoModelForCausalLM.from_pretrained(model_name)
```

`OUT:`

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[4], line 29
     25 model_load_tic = timeit.default_timer()
     27 #model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
---> 29 model = AutoModelForCausalLM.from_pretrained(model_name)
     30 model_load_toc = timeit.default_timer()
     32 model_load_duration = model_load_toc - model_load_tic

File ~\.conda\envs\rwkv-lora-pat\lib\site-packages\transformers\models\auto\auto_factory.py:566,
in _BaseAutoModelClass.from_pretrained(cls, pretrained_model_name_or_path, *model_args, **kwargs)
    562     model_class = _get_model_class(config, cls._model_mapping)
    563     return model_class.from_pretrained(
    564         pretrained_model_name_or_path, *model_args, config=config, **hub_kwargs, **kwargs
    565     )
--> 566 raise ValueError(
    567     f"Unrecognized configuration class {config.__class__} for this kind of AutoModel:
{cls.__name__}.\n"
    568     f"Model type should be one of {', '.join(c.__name__ for c in
cls._model_mapping.keys())}."
    569 )
```

```
ValueError: Unrecognized configuration class <class
'transformers.models.t5.configuration_t5.T5Config'> for this kind of AutoModel:
AutoModelForCausalLM.
Model type should be one of BartConfig, BertConfig, BertGenerationConfig, BigBirdConfig,
BigBirdPegasusConfig, BioGptConfig, BlenderbotConfig, BlenderbotSmallConfig, BloomConfig,
CamembertConfig, LlamaConfig, CodeGenConfig, CohereConfig, CpmAntConfig, CTRLConfig,
Data2VecTextConfig, DbrxConfig, ElectraConfig, ErnieConfig, FalconConfig, FuyuConfig, GemmaConfig,
GitConfig, GPT2Config, GPT2Config, GPTBigCodeConfig, GPTNeoConfig, GPTNeoXConfig,
GPTNeoXJapaneseConfig, GPTJConfig, JambaConfig, JetMoeConfig, LlamaConfig, MambaConfig,
MarianConfig, MBartConfig, MegaConfig, MegatronBertConfig, MistralConfig, MixtralConfig,
MptConfig, MusicgenConfig, MusicgenMelodyConfig, MvpConfig, OlmoConfig, OpenLlamaConfig,
OpenAIGPTConfig, OPTConfig, PegasusConfig, PersimmonConfig, PhiConfig, Phi3Config, PLBartConfig,
ProphetNetConfig, QDQBertConfig, Qwen2Config, Qwen2MoeConfig, RecurrentGemmaConfig,
ReformerConfig, RemBertConfig, RobertaConfig, RobertaPreLayerNormConfig, RoCBertConfig,
RoFormerConfig, RwkvConfig, Speech2Text2Config, StableLmConfig, Starcoder2Config, TransfoXLConfig,
TrOCRConfig, WhisperConfig, XGLMConfig, XLMConfig, XLMProphetNetConfig, XLMRobertaConfig,
XLMRobertaXLConfig, XLNetConfig, XmodConfig.
```

Trying some things I've been learning (architecture)

In [14]:
```
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 [15]:
```
model_arch_str = str(model)

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

Some other saves

In [16]:
```
pickle_filename = "lora_flan_t5_cpu_objects.pkl"
objects_to_pickle = []
objects_to_pickle.append(model_arch_str)
```

Prompt and Trainer - minimize

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 [17]:
```
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.

Debugging strange results note

Taking out the `run_name` and `overwrite_output_dir` arguments for the `Training_Arguments`. If we still have a problem, I'll also take out the `logging_strategy` and `logging_steps`. I hope I don't have to do the latter.

In [18]:
```
!powershell -c (Get-Date -UFormat \"%s_%Y%m%dT%H%M%S%Z00\") -replace '[.][0-9]*_', '_'
```

```
1717845924_20240608T112524-0600
```

Output was:

```
timestamp
```

Okay, here are the args and config

This next cell has the code which might be causing the `trainer` code not to execute. I want it to execute, though it might still give wrong answers.

cf. Section Trainer - the Actual Trainer Part

In [19]:
```
#  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='steps',
                    eval_steps=50,
                    hub_model_id="dwb-flan-t5-small-lora-finetune",
                    run_name="dwb-flan-samsum-run-cpu-20240609-01",
                    #  has nodename (machine), when this param is
                    #+ unset
                    save_strategy='steps',
                    save_steps=50,
                    overwrite_output_dir=True,
                    log_level='info',
                    logging_dir='logging',
                    logging_strategy='steps',
                    logging_first_step=True,
                    logging_steps=50,
)

#  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',
)
```

Details on `task_type` - minimize

`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.
```

Details on `TrainingArguments` - minimize

Documentation is at:

https://huggingface.co/docs/transformers/main/en/main_classes/trainer#transformers.TrainingArguments

archived at

https://web.archive.org/web/20240608164657/https://huggingface.co/docs/transformers/main/en/main_classes/trainer#transformers.Tr

The actual file (from which the docs come) is at the following path on my computer

`C:\Users\bballdave025\.conda\envs\rwkv-lora-pat\Lib\site-packages\transformers\training_args.py`

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 [20]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y%m%dT%H%M%S%Z00\") -replace '[.][0-9]*_', '_'
```

```
1717845925_20240608T112525-0600
```

Output was:

`timestamp`

In [21]:
```
system_info_as_script.run(do_network_info=False)
```

```
##########################  System Information  ##########################
System: Windows
Node Name: MYMACHINE
Release: 10
Version: 10.0.19045
Machine: AMD64
Processor: Intel64 Family 6 Model 165 Stepping 3, GenuineIntel
Processor: Intel(R) Core(TM) i3-10100 CPU @ 3.60GHz
Ip-Address: NOT-FOR-NOW
Mac-Address: NOT-FOR-NOW

############################  Boot Time  ############################
Boot Time (date and time of last boot) was
Boot Time: 2024-6-7T13:58:21

############################  CPU Info  ############################
Physical cores: 4
Total cores: 8
CPU Usage Per Core:
Core 0: 1.6%
Core 1: 0.0%
Core 2: 3.1%
Core 3: 0.0%
Core 4: 1.6%
Core 5: 0.0%
Core 6: 1.6%
Core 7: 0.0%
Total CPU Usage: 20.6%
Max Frequency: 3600.00Mhz
Min Frequency: 0.00Mhz
Current Frequency: 3600.00Mhz

############################  GPU Info  ############################
Information on GPU(s)/Graphics Card(s)
 (if any such information is to be found)


Using  PyTorch  and the `torch.cuda.is_available()` method.
The statement, 'There is CUDA and an appropriate GPU',
  is ...  False

Using  TensorFlow  with several of its methods.
```

```
  Attempting to get GPU Device List
No GPU devices found by TensorFlow.
# Tensorflow can give us CPU (and/or GPU) info.
The info here might help you know if we're running on a CPU.

Trying to use some nvidia code ( nvidia-smi ) to find information
  That nvidia stuff didn't work
  The error information is:
[WinError 2] The system cannot find the file specified
  Neither a big surprise nor a big deal
  That's the end of the nvidia try.

Trying to use  [py]lspci   to find information
   ... problem with pylspci.parsers.SimpleParser
      ... not especially surprising, nor a big deal
Command '['lspci', '-mm', '-nn']' returned non-zero exit status 1.
#  Note, if it said something about the file not being
#+ found, 'the file' is `lscpi`
         ... continuing with other ways to get information

There might be graphics card/GPU information in
the output from a python-only screenfetch copycat.
Getting the 'GPU' line from the qwqfetch output
GPU: Trigger 6 External Graphics

Those are all our chances to find out about any GPU/Graphics Cards

######################  Memory (RAM) Information  ######################
Total: 31.67GiB
Available: 22.43GiB
Used: 9.24GiB
Percentage: 29.2%
      =============== SWAP Memory ===============
Total: 4.75GiB
Free: 4.75GiB
Used: 0.00iB
Percentage: 0.0%

############################  Disk Info  ############################
Partitions and Usage:
=== Device: C:\ ===
  Mountpoint: C:\
```

```
  File system type: NTFS
  Total Size: 915.94GiB
  Used: 595.49GiB
  Free: 320.45GiB
  Percentage: 65.0%
=== Device: D:\ ===
  Mountpoint: D:\
  File system type: exFAT
  Total Size: 12.73TiB
  Used: 1.99TiB
  Free: 10.75TiB
  Percentage: 15.6%
=== Device: E:\ ===
  Mountpoint: E:\
  File system type: FAT32
  Total Size: 115.31GiB
  Used: 46.08GiB
  Free: 69.23GiB
  Percentage: 40.0%
  Since last boot,
Total read: 8.69GiB
Total write: 5.86GiB

### No network info for now ###

############################################################
##### The last attempts for any useful system info #####
These might give lots of info, or new info.
Then again, they might fail miserably.

Try to access lscpi (through python).
   ... problem with pylspci.parsers.SimpleParser
      ... not especially surprising, nor a big deal
Command '['lspci', '-mm', '-nn']' returned non-zero exit status 1.
#  Note, if it said something about the file not being
#+ found, 'the file' is `lscpi`
         ... continuing with other ways to get information

From qwqfetch, a python-only copycat of screenfetch
Should be a good recap, but might not be complete.
 ...
bballdave025@MYMACHINE
```

```
---------------------
OS: Microsoft Windows 10 Home AMD64
Host: Dell Inc. 0KV3RP
Kernel: NT 10.0.19045
Uptime: 21 hours, 27 mins
Shell: jupyter-notebook
Resolution: 1920x1080
DE: Windows Shell
Terminal: jupyter-notebook
CPU: Intel Core i3-10100 @ 3.60GHz (4) @ 3.60 GHz
GPU: Trigger 6 External Graphics
Memory: 9.26 GiB / 31.67 GiB

That's all we've got.
```

The reboot worked. I ran this from an elevated command prompt a while ago. The result are in the file,

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

ROUGE Metrics - minimize

Some references from the Google Research 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 incl SkipGrams but Not Using Now - minimize

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]:
```
do_show_dwb_rouge_doc = False
```

Documentation for my methods - minimize

In [23]:
```
#import dwb_rouge_scores # done with all the other imports
```

```
if do_show_dwb_rouge_doc:
    sep_banner_1 = "  " + "#" + "+"*60 + "#"
    sep_banner_2 = "     " + "#" + "~"*30 + "#"
    print()
    print()
    print(sep_banner_1)
    help(dwb_rouge_scores.dwb_rouge_n)
    print()
    print()
    print(sep_banner_1)
    print()
    print()
    help(dwb_rouge_scores.dwb_rouge_L)
    print()
    print(sep_banner_2)
    print()
    print("dwb_rouge_L needs dwb_lcs")
    print()
    print(sep_banner_2)
    print()
    help(dwb_rouge_scores.dwb_lcs)
    print()
    print()
    print(sep_banner_1)
    print()
    print()
    help(dwb_rouge_scores.dwb_rouge_s)
    print()
    print(sep_banner_2)
    print()
    print("dwb_rouge_s needs dwb_skipngrams")
    print()
    print(sep_banner_2)
    print()
    help(dwb_rouge_scores.dwb_skipngrams)
    print()
    print()
    print(sep_banner_1)
    print()
    print()
    help(dwb_rouge_scores.dwb_rouge_Lsum)
    print()
```

```
    print(sep_banner_2)
    print()
    print("dwb_rouge_Lsum just wraps google-research's rouge_score's")
    print("(from `pip install rouge-score`) version of rougeLsum")
    print()
    print()
    print(sep_banner_1)
##endof:  if do_show_dwb_rouge_doc
```

Other useful ROUGE code - Run/Evaluate Code Even if You'll Hide It

(found and created as I go along)

In [24]:
```
def format_rouge_score_rough(this_rouge_str):
    '''

    '''

    rouge_ret_str = this_rouge_str

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

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

    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 [25]:
```
def print_rouge_scores(result, sample_num_or_header=None):
    '''

    '''

    print("\n\n---------- ROUGE SCORES ----------")
    if sample_num_or_header is None:
        print("  --------- dialogue ----------")
    elif type(sample_num_or_header) is int:
        print(f"  --------- dialogue {sample_num_or_header+1} " + \
              "----------")
    else:
        print(f"  --------- {sample_num_or_header} ----------")
    ##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 [26]:
```
#-------------------------------------------------------------------------------
# #  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 from
    the googleresearch/rouge method it claims to be calling.
    '''

    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

    result = "there-is-some-problem-" + \
             "in-compute_google_rouge_score"
                                    #  scoping (if we weren't
                                    #+ in Python) and having
```

```
                                    #+ a sort of error message

    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 use_aggregator

    return result

##endof:  compute_google_rouge_score
```

Find specific dialog - minimize

I found a nice, short, interesting conversation while doing the random summaries, so I went and found its index.

In [27]:
```
tic = timeit.default_timer()

str_to_find = "Damien: Omg..I'm glad Sunday is only once a week"

for sample_num in range(len(dataset['test'])):
    this_sample = dataset['test'][sample_num]
    this_dialogue = this_sample['dialogue']
    if str_to_find in this_dialogue:
        print(f"sample_num: {sample_num}")
        print(f"this_dialogue: \n{this_dialogue}")
        print()
        print("this_sample:")
        print(str(this_sample))
        print()
    ##endof:  if str_to_find in this_dialogue
##endof:  for sample_number in range(len(dataset))

toc = timeit.default_timer()

print("Finding the sample in the test dataset (well,")
print("actually looking at every sample in the test")
print("dataset, regardless of whether we had found")
```

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

my_duration = toc - tic

elapsed_time_str = format_timespan(my_duration)

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

print()

print(f"Total size of test dataset: {sample_num}")
```

```
sample_num: 224
this_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.

this_sample:
{'dialogue': "Abigail: It's Sundaay.\nDamien: So?..\nAbigail: You know what that means.\nDamien: Hmm no I don't x)\nA
bigail: Sunday means we go to church~.\nDamien: Oh, yeah..\nAbigail: Don't forget to put on a coat and tie.\nDamien:
A coat and tie?.. Why?\nAbigail: To show respect to God and others.\nDamien: Omg..I'm glad Sunday is only once a wee
k.\nAbigail: I hope God didn't hear that.\nDamien: He'll forgive me 😇\nAbigail: Just be ready on time please.", 'i
d': '13681509', 'summary': 'Abigail and Damien are going to church on Sunday. Damien has to put on a coat and tie.'}

Finding the sample in the test dataset (well,
actually looking at every sample in the test
dataset, regardless of whether we had found
something.
took 0.0464 seconds.
which equates to 0.05 seconds

Total size of test dataset: 818
```

The interesting conversation

In [28]:
```python
my_index = 224
my_complete_entry = dataset['test'][my_index]
my_cool_str = dataset['test'][my_index]['dialogue']
print(my_cool_str)
objects_to_pickle.append(my_cool_str)
my_cool_list = [f"my_index: {my_index}", my_cool_str, my_complete_entry]
```

```
pprint.pp(my_cool_list)
objects_to_pickle.append(my_cool_list)
```

```
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.
['my_index: 224',
 "Abigail: It's Sundaay.\n"
 'Damien: So?..\n'
 'Abigail: You know what that means.\n'
 "Damien: Hmm no I don't x)\n"
 'Abigail: Sunday means we go to church~.\n'
 'Damien: Oh, yeah..\n'
 "Abigail: Don't forget to put on a coat and tie.\n"
 'Damien: A coat and tie?.. Why?\n'
 'Abigail: To show respect to God and others.\n'
 "Damien: Omg..I'm glad Sunday is only once a week.\n"
 "Abigail: I hope God didn't hear that.\n"
 "Damien: He'll forgive me 😇\n"
 'Abigail: Just be ready on time please.',
 {'dialogue': "Abigail: It's Sundaay.\n"
              'Damien: So?..\n'
              'Abigail: You know what that means.\n'
              "Damien: Hmm no I don't x)\n"
              'Abigail: Sunday means we go to church~.\n'
              'Damien: Oh, yeah..\n'
              "Abigail: Don't forget to put on a coat and tie.\n"
              'Damien: A coat and tie?.. Why?\n'
              'Abigail: To show respect to God and others.\n'
              "Damien: Omg..I'm glad Sunday is only once a week.\n"
              "Abigail: I hope God didn't hear that.\n"
              "Damien: He'll forgive me 😇\n"
              'Abigail: Just be ready on time please.',
  'id': '13681509',
```

```
 'summary': 'Abigail and Damien are going to church on Sunday. Damien has to '
            'put on a coat and tie.'}]
```

Let's get the sizes of all parts of the dataset

In [29]:
```
size_of_train = len(dataset['train'])
size_of_eval = len(dataset['evaluation'])
size_of_test = len(dataset['test'])

print(f"size_of_train : {size_of_train}")
print(f"size_of_eval  : {size_of_eval}")
print(f"size_of_test  : {size_of_test}")
```

```
size_of_train : 14732
size_of_eval  : 818
size_of_test  : 819
```

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

In [30]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y%m%dT%H%M%S%Z00\") -replace '[.][0-9]*_', '_'
```

```
1717845933_20240608T112533-0600
```

Output was:

`timestamp`

Just one summarization to begin with, randomly picked

Well, not so randomly, anymore

In [31]:
```
#  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.
#
#  For sharing with Pat, I'm making it repeatable
```

```
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 kids 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 have sex and wait for a baby while
having a normal life. Not getting fat, having the same 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 money and she couldn't even take th
em 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 pregnant. 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 pregnant
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 have a good job, I'm financially ind
ependent. 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 pregn
ant woman dropped some money and couldn't take them from the floor. She's about to give birth.
```

Now, a couple summarizations with comparisons to ground truth

In [32]:
```
summarizer = pipeline('summarization',
                      model=model,
                      tokenizer=tokenizer)

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 later loop, i.e. with lists
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 summarization task, where outputs s
horter than the input are typically wanted, you might 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 Larry well. Larry called her last tim
e they were at a park. Hannah will text Larry.
```

```
C:\Users\Anast\AppData\Local\Temp\ipykernel_12056\725041399.py:28: FutureWarning: load_metric is deprecated and will
be removed in the next major version of datasets. Use 'evaluate.load' instead, from the new library 🤗 Evaluate: http
s://huggingface.co/docs/evaluate
  rouge = load_metric('rouge', trust_remote_code=True)
```

In [33]:
```
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=0.06666666666666667,
          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 [34]:

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

#  I don't want to aggregate, yet.
pred_test_list = []
ref_test_list = []

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}")

ref_test_list.append(ground_summary)
pred_test_list.append(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 summarization task, where outputs s
horter than the input are typically wanted, you might 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 tie.
flan-t5-small summary:
Abigail, Damien and Damien go to church on Sunday. They are going to pray for God and others. Damien is glad Sunday i
s only once a week.
```

In [35]:
```
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 - minimize

```
    # @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]*_', '_'
```

```
1717845939_20240608T112539-0600
```

Output was:

`timestamp`

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 on Complete Test Set

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

```
1717845940_20240608T112540-0600
```

Output was:

`timestamp`

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

In [39]:
```
do_have_lotta_output_from_all_dialogs_summaries_1 = False
```

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

In [40]:
```
print("That last boolean has the value:")
print(f"{do_have_lotta_output_from_all_dialogs_summaries_1}")
```

```
That last boolean has the value:
False
```

There could be up to megabytes worth of text output if you've changed it to `True`.

In [41]:
```
#  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)

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

    baseline_sample_dialog_list.append(this_sample['dialogue'])
    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 (doing the baseline)")
print(f"took {baseline_toc - baseline_tic:0.4f} seconds.")

baseline_time_str = format_timespan(baseline_duration)

print(f"which equates to {baseline_time_str}")
```

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

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

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

objects_to_pickle.append(baseline_sample_dialog_list)
objects_to_pickle.append(baseline_prediction_list)
objects_to_pickle.append(baseline_reference_list)
objects_to_pickle.append(baseline_results)
```

```
Getting things ready for scoring (doing the baseline)
took 1100.4034 seconds.
which equates to 18 minutes and 20.4 seconds
```

In [42]:
```
##  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 [43]:
```
print_rouge_scores(baseline_results, "BASELINE")
```

```
---------- ROUGE SCORES ----------
  --------- BASELINE ----------
ROUGE-1 results
AggregateScore(
     low=Score(
          precision=0.36323792796254495,
          recall=0.5389572077652378,
          fmeasure=0.4121922114955201),
     mid=Score(
          precision=0.3733579418873792,
          recall=0.5519833607010016,
          fmeasure=0.42140038554194087),
     high=Score(
          precision=0.3841066527398303,
          recall=0.5654634115541827,
          fmeasure=0.43108998286468664)
)
ROUGE-2 results
AggregateScore(
     low=Score(
          precision=0.15925462454674427,
          recall=0.24407889616325276,
          fmeasure=0.18075696508371478),
     mid=Score(
          precision=0.16778790384209413,
          recall=0.25699792119566545,
          fmeasure=0.19019414340220459),
     high=Score(
          precision=0.1770395400263706,
          recall=0.2699836832354256,
          fmeasure=0.19971083652555854)
)
ROUGE-L results
AggregateScore(
     low=Score(
          precision=0.2804799948778764,
          recall=0.42293212107020284,
          fmeasure=0.32025419531533744),
     mid=Score(
          precision=0.28920324877812476,
          recall=0.43542261213145617,
```

```
          fmeasure=0.3280522451383458),
     high=Score(
          precision=0.2986030814587809,
          recall=0.44741373320548616,
          fmeasure=0.33701239118750953)
)
ROUGE-Lsum results
AggregateScore(
     low=Score(
          precision=0.28036218112473665,
          recall=0.42289802566338497,
          fmeasure=0.3193301330569866),
     mid=Score(
          precision=0.2892036903522128,
          recall=0.4351173994472087,
          fmeasure=0.3280497397975705),
     high=Score(
          precision=0.2981799105260594,
          recall=0.44799092741287305,
          fmeasure=0.3365370888024341)
)
```

Trainer - the Actual Trainer Part

In [44]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y-%m-%dT%H%M%S%Z00\") -replace '[.][0-9]*_', '_'
```

```
1717847043_2024-06-08T114403-0600
```

Output was:

`timestamp`

This next cell has the code that I need to check. I want it to execute, though it might still give wrong answers.

cf. Section Trainer - the LoRA Setup Part

In [45]:
```
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,
                    max_seq_length=750
)
##  Warnings are below output.
```

```
WARNING:bitsandbytes.cextension:The installed version of bitsandbytes was compiled without GPU support. 8-bit optimizers, 8-bit multiplication, and GPU quantization are unavailable.
Generating train split: 0 examples [00:00, ? examples/s]
Generating train split: 0 examples [00:00, ? examples/s]
```

Warnings I'll Now Worry About - to fix CoLab

~~Warnings I Won't Worry About, Yet - don't minimize~~

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.

~~I never ran the updated cell, (with an additional parameter, `max_seq_length=675`), so the Warning and Advice are still there.~~

To try and get this working on CoLab, I'm going to use `max_seq_length=750` .

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

In [46]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y-%m-%dT%H%M%S%Z00\") -replace '[.][0-9]*_', '_'
```

```
1717847047_2024-06-08T114407-0600
```

Output was:

`timestamp`

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 [47]:
```
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.")
training_time_str = format_timespan(training_duration)
print(f"which equates to {training_time_str}")
```

```
***** Running training *****
  Num examples = 4,194
  Num Epochs = 1
  Instantaneous batch size per device = 4
  Total train batch size (w. parallel, distributed & accumulation) = 4
  Gradient Accumulation steps = 1
  Total optimization steps = 1,049
  Number of trainable parameters = 2,752,512
```

[1049/1049 4:00:52, Epoch 1/1]

Step	Training Loss	Validation Loss
250	0.160900	0.057427
500	0.095400	0.033200
750	0.084000	0.027997
1000	0.081900	0.026790

```
***** Running Evaluation *****
  Num examples = 228
  Batch size = 4
Saving model checkpoint to output\checkpoint-250
C:\Users\Anast\.conda\envs\rwkv-lora-pat\lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `res
ume_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 configuration file config.json from cache at C:\Users\Anast\.cache\huggingface\hub\models--google--flan-t5-sm
all\snapshots\0fc9ddf78a1e988dac52e2dac162b0ede4fd74ab\config.json
Model config T5Config {
  "architectures": [
    "T5ForConditionalGeneration"
  ],
  "classifier_dropout": 0.0,
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dense_act_fn": "gelu_new",
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": true,
  "is_gated_act": true,
  "layer_norm_epsilon": 1e-06,
  "model_type": "t5",
  "n_positions": 512,
  "num_decoder_layers": 8,
  "num_heads": 6,
  "num_layers": 8,
  "output_past": true,
  "pad_token_id": 0,
  "relative_attention_max_distance": 128,
  "relative_attention_num_buckets": 32,
  "task_specific_params": {
    "summarization": {
      "early_stopping": true,
      "length_penalty": 2.0,
      "max_length": 200,
      "min_length": 30,
```

```
      "no_repeat_ngram_size": 3,
      "num_beams": 4,
      "prefix": "summarize: "
    },
    "translation_en_to_de": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to German: "
    },
    "translation_en_to_fr": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to French: "
    },
    "translation_en_to_ro": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to Romanian: "
    }
  },
  "tie_word_embeddings": false,
  "transformers_version": "4.41.1",
  "use_cache": true,
  "vocab_size": 32128
}

tokenizer config file saved in output\checkpoint-250\tokenizer_config.json
Special tokens file saved in output\checkpoint-250\special_tokens_map.json
***** Running Evaluation *****
  Num examples = 228
  Batch size = 4
Saving model checkpoint to output\checkpoint-500
C:\Users\Anast\.conda\envs\rwkv-lora-pat\lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `res
ume_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 configuration file config.json from cache at C:\Users\Anast\.cache\huggingface\hub\models--google--flan-t5-sm
all\snapshots\0fc9ddf78a1e988dac52e2dac162b0ede4fd74ab\config.json
Model config T5Config {
```

```
  "architectures": [
    "T5ForConditionalGeneration"
  ],
  "classifier_dropout": 0.0,
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dense_act_fn": "gelu_new",
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": true,
  "is_gated_act": true,
  "layer_norm_epsilon": 1e-06,
  "model_type": "t5",
  "n_positions": 512,
  "num_decoder_layers": 8,
  "num_heads": 6,
  "num_layers": 8,
  "output_past": true,
  "pad_token_id": 0,
  "relative_attention_max_distance": 128,
  "relative_attention_num_buckets": 32,
  "task_specific_params": {
    "summarization": {
      "early_stopping": true,
      "length_penalty": 2.0,
      "max_length": 200,
      "min_length": 30,
      "no_repeat_ngram_size": 3,
      "num_beams": 4,
      "prefix": "summarize: "
    },
    "translation_en_to_de": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to German: "
    },
    "translation_en_to_fr": {
```

```
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to French: "
    },
    "translation_en_to_ro": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to Romanian: "
    }
  },
  "tie_word_embeddings": false,
  "transformers_version": "4.41.1",
  "use_cache": true,
  "vocab_size": 32128
}

tokenizer config file saved in output\checkpoint-500\tokenizer_config.json
Special tokens file saved in output\checkpoint-500\special_tokens_map.json
***** Running Evaluation *****
  Num examples = 228
  Batch size = 4
Saving model checkpoint to output\checkpoint-750
C:\Users\Anast\.conda\envs\rwkv-lora-pat\lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `res
ume_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 configuration file config.json from cache at C:\Users\Anast\.cache\huggingface\hub\models--google--flan-t5-sm
all\snapshots\0fc9ddf78a1e988dac52e2dac162b0ede4fd74ab\config.json
Model config T5Config {
  "architectures": [
    "T5ForConditionalGeneration"
  ],
  "classifier_dropout": 0.0,
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dense_act_fn": "gelu_new",
  "dropout_rate": 0.1,
  "eos_token_id": 1,
```

```
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": true,
  "is_gated_act": true,
  "layer_norm_epsilon": 1e-06,
  "model_type": "t5",
  "n_positions": 512,
  "num_decoder_layers": 8,
  "num_heads": 6,
  "num_layers": 8,
  "output_past": true,
  "pad_token_id": 0,
  "relative_attention_max_distance": 128,
  "relative_attention_num_buckets": 32,
  "task_specific_params": {
    "summarization": {
      "early_stopping": true,
      "length_penalty": 2.0,
      "max_length": 200,
      "min_length": 30,
      "no_repeat_ngram_size": 3,
      "num_beams": 4,
      "prefix": "summarize: "
    },
    "translation_en_to_de": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to German: "
    },
    "translation_en_to_fr": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to French: "
    },
    "translation_en_to_ro": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to Romanian: "
    }
```

```
  },
  "tie_word_embeddings": false,
  "transformers_version": "4.41.1",
  "use_cache": true,
  "vocab_size": 32128
}

tokenizer config file saved in output\checkpoint-750\tokenizer_config.json
Special tokens file saved in output\checkpoint-750\special_tokens_map.json
***** Running Evaluation *****
  Num examples = 228
  Batch size = 4
Saving model checkpoint to output\checkpoint-1000
C:\Users\Anast\.conda\envs\rwkv-lora-pat\lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `res
ume_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 configuration file config.json from cache at C:\Users\Anast\.cache\huggingface\hub\models--google--flan-t5-sm
all\snapshots\0fc9ddf78a1e988dac52e2dac162b0ede4fd74ab\config.json
Model config T5Config {
  "architectures": [
    "T5ForConditionalGeneration"
  ],
  "classifier_dropout": 0.0,
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dense_act_fn": "gelu_new",
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": true,
  "is_gated_act": true,
  "layer_norm_epsilon": 1e-06,
  "model_type": "t5",
  "n_positions": 512,
  "num_decoder_layers": 8,
  "num_heads": 6,
  "num_layers": 8,
  "output_past": true,
```

```
  "pad_token_id": 0,
  "relative_attention_max_distance": 128,
  "relative_attention_num_buckets": 32,
  "task_specific_params": {
    "summarization": {
      "early_stopping": true,
      "length_penalty": 2.0,
      "max_length": 200,
      "min_length": 30,
      "no_repeat_ngram_size": 3,
      "num_beams": 4,
      "prefix": "summarize: "
    },
    "translation_en_to_de": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to German: "
    },
    "translation_en_to_fr": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to French: "
    },
    "translation_en_to_ro": {
      "early_stopping": true,
      "max_length": 300,
      "num_beams": 4,
      "prefix": "translate English to Romanian: "
    }
  },
  "tie_word_embeddings": false,
  "transformers_version": "4.41.1",
  "use_cache": true,
  "vocab_size": 32128
}

tokenizer config file saved in output\checkpoint-1000\tokenizer_config.json
Special tokens file saved in output\checkpoint-1000\special_tokens_map.json
```

```
Training completed. Do not forget to share your model on huggingface.co/models =)
```

```
tic: 78350.572271
toc: 92815.6876683
Training took 14465.1154 seconds.
which equates to 4 hours, 1 minute and 5.12 seconds
```

In [48]:
```
# # Don't need this again
!powershell -c (Get-Date -UFormat \"%s_%Y-%m-%dT%H%M%S%Z00\") -replace '[.][0-9]*_', '_'
```

```
1717861513_2024-06-08T154513-0600
```

Output was:

`timestamp`

Thinking about it and learning - minimize

@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?

Update: Answer is below.

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

Including four additional parameters - those at the end

```
    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",
                        run_name="dwb-flan-samsum-run-cpu-20240607-01",
                        #  has nodename (machine), when this param is
                        #+ unset
                        overwrite_output_dir=True,
                        logging_strategy='steps',
                        logging_steps=32,
    )
```

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:

`timestamp`

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')
  #  '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()
```

Hugging Face Repo Info - minimize

Part of the output included text giving 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 [ ]:
```
# Read token. Will bring up text entry to paste token string
notebook_login()
```

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

Output was:

```
timestamp
```

(If you have problems that note `data_files` or `dataset` or `prompt_instruction_format`, make sure that the cells where these are defined have been run, i.e. the kernel hasn't been restarted since they were initialized.)

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

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

In [ ]:
```
new_model_load_tic = timeit.default_timer()
new_model = AutoModelForSeq2SeqLM.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}")

new_tokenizer.pad_token = new_tokenizer.eos_token
new_tokenizer.padding_side = "right"

print()
print()

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

old_tokenizer_time_str = format_timespan(old_tokenizer_duration)

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

#  padding instructions for the tokenizer
old_tokenizer.pad_token = tokenizer.eos_token
old_tokenizer.padding_side = "right"
```

Stuff for model architecture - post-LoRA

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

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

with open(
  "dwb-flan-t5-small-lora-finetune.model-architecture.txt",
  'w',
  encoding='utf-8') as fhn:
    fhn.write(new_model_arch_str)
##endof:  with open ... fhn

objects_to_pickle.append(new_model_arch_str)
```

@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.

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

Output was:

```
timestamp
```

Note on the `tokenizer` parameter in the `pipline` function - minimize

Note that in Guptal's example/tutorial, he did not give a `tokenizer` parameter to the `summarizer = pipeline` code. I'm trying that quickly.

Consistency in values of model and tokenizer

In [ ]:
```
# If we want to keep it consistent, use these. If not, change them at will.
model_to_use = new_model
tokenizer_to_use = new_tokenizer
```

Try one picked at random

Well, not so randomly, anymore

In [ ]:
```
#  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.
#
#  User repeatability when sharing with Pat


do_seed_for_repeatable = True

summarizer = pipeline('summarization',
                      model=new_model) #,
```

```
                    #tokenizer=new_tokenizer)

#  Look down at the section, 'Notes on investigations to fix weirdness'

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']}")
```

Notes on investigations to fix weirdness

@todo : format this more nicely

```
    ## Trials to fix weirdness.
    ## model=old_model, tokenizer=old_tokenizer : matches baseline
    ##                                            (quick)
    ## model=new_model, tokenizer=new_tokenizer : weird results
    ##                                            (takes significantly longer, too)
    ## model=new_model, tokenizer=old_tokenizer : weird results
    ##                                            (takes significantly longer, too)
    ## model=old_model, tokenizer=new_tokenizer : actually matches baseline, which
    ##                                            would seem to require a change in
    ##                                            hypothesis as to why the
    ##                                            weirdness and longer inference are
    ##                                            happening. (Likely not tokenizer.)
    ##                                            (quick)
    ##
    ##  I had thought that doing 'old_model' and 'new_tokenizer' gave me weird
    ##+ results, too. Good thing to come back and check things.
    ##  Still, the training results show the model was learning and improving.
```

Now, a couple summarizations with comparison to ground truth

In [ ]:
```
summarizer = pipeline('summarization',
                      model=new_model) #,
                      #tokenizer=new_tokenizer)

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
)
# testing `compute_google_rouge_score` to `replace rouge.compute`

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

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

In [ ]:
```
summarizer = pipeline('summarization',
                      model=new_model) #,
                      #tokenizer=new_tokenizer)

# I don't want to aggregate, yet
pred_test_list = []
ref_test_list = []

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:\n{res_summary}")

ref_test_list.append(ground_summary)
pred_test_list.append(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
)
```

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

Evaluation on the Test Set and Comparison to Baseline

Verbosity stuff - get rid of the nice advice

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

Output was:

```
timestamp
```

In [ ]:
```
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}")
```

Here's the actual evaluation

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

Output was:

`timestamp`

!!! 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 [ ]:
```
do_have_lotta_output_from_all_dialogs_summaries_2 = False
```

Are you sure about the value of that last boolean? 2

In [ ]:
```
print("That last boolean has the value:")
print(f"{do_have_lotta_output_from_all_dialogs_summaries_2}")
```

There could be up to megabytes worth of text output if you've changed it to `True`.

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

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

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_2:
        print("="*75)
        print(f"dialogue: \n{this_sample['dialogue']}\n---------------")
    ##endof:  if do_have_lotta_output_from_all_dialogs_summaries_2

    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_2:
        print("-"*70)
        print(f"human-genratd summary:\n{ground_tat_summary}")
        print("-"*70)
        print(f"flan-t5-small summary:\n{res_tat_summary}")
        print("-"*70)
    ##endof:  if do_have_lotta_output_from_all_dialogs_summaries_2

    tat_sample_dialog_list.append(this_sample['dialogue'])
    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']

objects_to_pickle.append(tat_sample_dialog_list)
objects_to_pickle.append(prediction_tat_list)
objects_to_pickle.append(reference_tat_list)
objects_to_pickle.append(results_tat)
```

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

Any comparison

In [ ]:
```
# any comparison code
```

Pickle things to pickle save

In [ ]:
```
objects_to_pickle_var_names = []

objects_to_pickle_var_names.append('model_arch_str')
objects_to_pickle_var_names.append('baseline_sample_dialog_list')
objects_to_pickle_var_names.append('baseline_prediction_list')
objects_to_pickle_var_names.append('baseline_reference_list')
objects_to_pickle_var_names.append('baseline_results')
objects_to_pickle_var_names.append('new_model_arch_str')
```

```
objects_to_pickle_var_names.append('tat_sample_dialog_list')
objects_to_pickle_var_names.append('prediction_tat_list')
objects_to_pickle_var_names.append('reference_tat_list')
objects_to_pickle_var_names.append('results_tat')
objects_to_pickle_var_names.append('objects_to_picle_var_names')

objects_to_pickle.append(objects_to_pickle_var_names)

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 - minimize

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/model_doc/rwkv