Deep Learning Project

By Victoria Lassner DSML 4220

Goal: Fine tune a model for abstractive Summarization.

Model: T5-Base with its Tokenizer

Websites: https://huggingface.co/docs/transformers/tasks/summarization

Future Models to Compare:

https://wandb.ai/mostafaibrahim17/ml-articles/reports/Fine-Tuning-LLaMa-2-for-Text-Summarization--Vmlldzo2NjA1OTAy

https://wandb.ai/mostafaibrahim17/ml-articles/reports/Crafting-Superior-Summaries-The-ChatGPT-Fine-Tuning-Guide--Vmlldzo1Njc5NDI1

Definitions:

Abstractive summarization = oncise summary of a text by understanding its meaning and creating new sentences, rather than simply extracting phrases from the original text.

Dataset: CNN/DailyMail: https://paperswithcode.com/dataset/cnn-daily-mail-1 BillSum

```
# disables weights and biases
import os
os.environ["WANDB_DISABLED"] = "true"
```

```
# downloads packages for model, dataset and tokenzier
# --Quiet limits output of messages
!pip install transformers datasets evaluate sentencepiece rouge_score --quiet
```

```
# Download packages
from datasets import load_dataset, concatenate_datasets
from transformers import T5ForConditionalGeneration, TrainingArguments, Trainer, T5Tokeni
import torch
from torch.utils.data import DataLoader
import torch
```

```
# Load CNN/Daily Mail Dataset from dataset package
# Limit samples to 7000 total.

train_sample_limit = 5000
val_sample_limit = 2000
```

```
dataset = load_dataset("cnn_dailymail", "3.0.0")
limited_train_data = dataset["train"].select(range(train_sample_limit))
limited_val_data = dataset["validation"].select(range(val_sample_limit))
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarnir
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public mc
  warnings.warn(
```

```
# preprocess data for model
tokenizer = T5Tokenizer.from_pretrained("t5-base")

# limit length of input articles and output summary
max_input_length = 512
max_target_length = 250

chunk_size = 1000
```

```
# process text into tokens usin T5tokenizer
def preprocess(examples):
    inputs = ["summarize: " + doc for doc in examples["article"]]
    targets = examples["highlights"]

    model_inputs = tokenizer(
        inputs,
        max_length=max_input_length,
        truncation=True,
        padding="max_length"
    )

    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
            targets,
            max_length=max_target_length,
            truncation=True,
            padding="max_length"
        )

    # Replace pad token with -100 to ignore in loss
    # Previously had errors due to padding with rouge eval
    labels["input_ids"] = [
      [(label if label != tokenizer.pad_token_id else -100) for label in label_seq]
      for label_seq in labels["input_ids"]
    ]

    model_inputs["labels"] = labels["input_ids"]
```

```
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

# break up data into chunks for easier training
def process_in_chunks(dataset, chunk_size, preprocess_fn):
    total_len = len(dataset)
    processed_chunks = []

    for i in range(0, total_len, chunk_size):
        chunk = dataset.select(range(i, min(i + chunk_size, total_len)))
        processed_chunk = chunk.map(
            preprocess_fn,
            batched=True,
            remove_columns=["article", "highlights", "id"]
        )
        processed_chunks.append(processed_chunk)

    return concatenate_datasets(processed_chunks)

# process the training and validation data into chunks
train_dataset = process_in_chunks(limited_train_data, chunk_size, preprocess)
val_dataset = process_in_chunks(limited_val_data, chunk_size, preprocess)
```

```
You are using the default legacy behaviour of the <class 'transformers.models.t5.toke
Map: 100%                                                  1000/1000 [00:11<00:00, 86.34 examples/
                                                                                                  s]
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py:3980:
  warnings.warn(
Map: 100%                                                  1000/1000 [00:04<00:00, 226.98 examples/
                                                                                                  s]
Map: 100%                                                  1000/1000 [00:05<00:00, 188.34 examples/
                                                                                                  s]
Map: 100%                                                  1000/1000 [00:04<00:00, 223.64 examples/
                                                                                                  s]
```

```
#Adds ROUGE to properly evaluate model
import evaluate
import numpy as np

rouge = evaluate.load("rouge")

def compute_metrics(eval_preds):
    preds, labels = eval_preds

    # If preds are logits, convert to token IDs
    if isinstance(preds, tuple):
```

```
        preds = preds[0]

    if preds.ndim == 3:  # logits
        preds = np.argmax(preds, axis=-1)

    # clip token IDs to vocab size
    preds = np.clip(preds, 0, tokenizer.vocab_size - 1)

    decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    result = rouge.compute(predictions=decoded_preds, references=decoded_labels, use_stem
    return {k: round(v * 100, 2) for k, v in result.items()}


# Load model T5-base
model = T5ForConditionalGeneration.from_pretrained("t5-base")

from transformers import Seq2SeqTrainingArguments

training_args = Seq2SeqTrainingArguments(
    output_dir="./t5-cnn-model",
    eval_steps=500,
    per_device_train_batch_size=4,    #batch size for training
    per_device_eval_batch_size=4,     #batch size for evaluation
    predict_with_generate=True,       #decoding
    generation_max_length=128,        #max tokens for generated sequences during eval/pre
    logging_steps=100,                #reports loss every 100 steps
    save_steps=1000,                  #saves model every 1000 steps
    num_train_epochs=3,
    fp16=True                          #GPU
)
```

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in
```

```
# adds padding so shorter sequences match the longest one
from transformers import DataCollatorForSeq2Seq

data_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)


# train model using hugging face's trainer class
from transformers import Seq2SeqTrainer

trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset
```

```
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
trainer.train()
```

```
<ipython-input-9-dd2b587ad4d3>:4: FutureWarning: `tokenizer` is deprecated and will b
  trainer = Seq2SeqTrainer(
Passing a tuple of `past_key_values` is deprecated and will be removed in Transformer
```

[3750/3750 31:31, Epoch 3/3]

Step	Training Loss
100	1.734000
200	1.629400
300	1.631200
400	1.662200
500	1.590900
600	1.598300
700	1.618200
800	1.612200
900	1.611600
1000	1.608600
1100	1.558500
1200	1.630200
1300	1.555100
1400	1.458100
1500	1.450800
1600	1.497700
1700	1.511900
1800	1.484700
1900	1.436800
2000	1.489600
2100	1.484800
2200	1.485900
2300	1.448900
2400	1.504600

2500	1.443900
2600	1.416300
2700	1.375300
2800	1.379100
2900	1.377600
3000	1.389600
3100	1.349000
3200	1.362700
3300	1.409300
3400	1.452300
3500	1.405300
3600	1.386000
3700	1.403400

```
TrainOutput(global_step=3750, training_loss=1.497725351969401,
metrics={'train_runtime': 1893.5016, 'train_samples_per_second': 7.922,
'train_steps_per_second': 1.98, 'total_flos': 9134368358400000.0, 'train_loss':
1.497725351969401, 'epoch': 3.0})
```

```
# evaluate model using ROUGE
metrics = trainer.evaluate()
print(metrics)
```

[500/500 15:57]

```
{'eval_loss': 1.878161907196045, 'eval_rouge1': 37.46, 'eval_rouge2': 16.15, 'eval_rc
```

```
#saves current state of model and tokenzier locally
model.save_pretrained("/content/t5_cnn_model_base_v4")
tokenizer.save_pretrained("/content/t5_cnn_model_base_v4")
```

```
('/content/t5_cnn_model_base_v4/tokenizer_config.json',
 '/content/t5_cnn_model_base_v4/special_tokens_map.json',
 '/content/t5_cnn_model_base_v4/spiece.model',
 '/content/t5_cnn_model_base_v4/added_tokens.json')
```

```
# save model to huggingface for easier access for web app
from huggingface_hub import HfApi, HfFolder
from transformers import AutoModelForSequenceClassification, AutoTokenizer
from huggingface_hub import login

login()
```

```
# Load model and tokenizer
model = T5ForConditionalGeneration.from_pretrained("/content/t5_cnn_model_base_v4")
tokenizer = T5Tokenizer.from_pretrained("/content/t5_cnn_model_base_v4")

# Save to HuggingFace
model.push_to_hub("vlassner01/t5_cnn_model_base_v4")
tokenizer.push_to_hub("vlassner01/t5_cnn_model_base_v4")
```

```
olab_upload not found in /root/.cache/huggingface/stored

model.safetensors: 100%                                    892M/892M [00:25<00:00, 39.7MB/
                                                          s]

README.md: 100%                                            5.17k/5.17k [00:00<00:00, 505kB/
                                                          s]

CommitInfo(commit_url='https://huggingface.co/vlassner01/t5_cnn_model_base_v4/
```

```
# CHATGPT's work around for spiece.model not uploading to huggingface
!pip install --upgrade huggingface-hub


from huggingface_hub import login
login()

!mkdir -p /content/hf_tokenizer_upload
!cp -r /content/t5_cnn_model_base_v3/* /content/hf_tokenizer_upload/

from huggingface_hub import upload_file

repo_name = "vlassner01/t5_cnn_model_base_v4"

folder_path = '/content/hf_tokenizer_upload'

upload_file(
    path_or_fileobj=f"{folder_path}/spiece.model",  # Replace with actual file path
    path_in_repo="spiece.model",  # Path in the Hugging Face repo
    repo_id=repo_name,
    commit_message="Upload spiece.model"
)

upload_file(
    path_or_fileobj=f"{folder_path}/tokenizer_config.json",  # Replace with actual file pat
    path_in_repo="tokenizer_config.json",  # Path in the Hugging Face repo
    repo_id=repo_name,
    commit_message="Upload tokenizer_config.json"
)

upload_file(
    path_or_fileobj=f"{folder_path}/special_tokens_map.json",  # Replace with actual file p
```

```
    path_in_repo="special_tokens_map.json",  # Path in the Hugging Face repo
    repo_id=repo_name,
    commit_message="Upload special_tokens_map.json"
)

upload_file(
    path_or_fileobj=f"{folder_path}/tokenizer.json",  # Replace with actual file path
    path_in_repo="tokenizer.json",  # Path in the Hugging Face repo
    repo_id=repo_name,
    commit_message="Upload tokenizer.json"
)
```

```
Requirement already satisfied: huggingface-hub in /usr/local/lib/python3.11/dist-pack
ERROR: Operation cancelled by user
```

Copy a token from your Hugging Face tokens page and

paste it below.

Immediately click login after copying your token or it

might be stored in plain text in this notebook file.

Token:

Add token as git credential?

Login

Pro Tip: If you don't already have one, you can create

a dedicated 'notebooks' token with 'write' access, that

you can then easily reuse for all notebooks.

```
cp: cannot stat '/content/t5_cnn_model_base_v3/*': No such file or directory
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-1-b7aecc9fa554> in <cell line: 0>()
     20
     21 # Upload individual files to Hugging Face
---> 22 upload_file(
     23     path_or_fileobj=f"{folder_path}/spiece.model",  # Replace with actual
file path
     24     path_in_repo="spiece.model",  # Path in the Hugging Face repo
```

4 frames

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/_commit_api.py in
__post_init__(self)
    180             path_or_fileobj =
os.path.normpath(os.path.expanduser(self.path_or_fileobj))
```

Next steps: Explain error