In [2]:
```
!pip install transformers[sentencepiece] datasets sacrebleu rouge_score py7zr -q
```

```
□[K     |████████████████████████████████| 5.5 MB 39.2 MB/s
□[K     |████████████████████████████████| 451 kB 71.7 MB/s
□[K     |████████████████████████████████| 118 kB 89.5 MB/s
□[K     |████████████████████████████████| 65 kB 4.4 MB/s
□[K     |████████████████████████████████| 182 kB 89.7 MB/s
□[K     |████████████████████████████████| 115 kB 87.5 MB/s
□[K     |████████████████████████████████| 212 kB 75.9 MB/s
□[K     |████████████████████████████████| 127 kB 87.1 MB/s
□[K     |████████████████████████████████| 357 kB 95.7 MB/s
□[K     |████████████████████████████████| 50 kB 6.7 MB/s
□[K     |████████████████████████████████| 2.3 MB 74.0 MB/s
□[K     |████████████████████████████████| 379 kB 87.1 MB/s
□[K     |████████████████████████████████| 138 kB 76.8 MB/s
□[K     |████████████████████████████████| 93 kB 2.4 MB/s
□[K     |████████████████████████████████| 7.6 MB 78.7 MB/s
□[K     |████████████████████████████████| 1.3 MB 89.9 MB/s
□[?25h  Building wheel for rouge-score (setup.py) ... □[?25l□[?25hdone
```

In [3]:
```
!nvidia-smi
```

```
Tue Nov 29 04:03:54 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   39C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

In [4]:
```
from transformers import pipeline, set_seed

import matplotlib.pyplot as plt
from datasets import load_dataset
import pandas as pd
from datasets import load_dataset, load_metric

from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

import nltk
from nltk.tokenize import sent_tokenize

from tqdm import tqdm
import torch

nltk.download("punkt")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out [4]: True

In [5]:
```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

device = "cuda" if torch.cuda.is_available() else "cpu"

model_ckpt = "google/pegasus-cnn_dailymail"

tokenizer = AutoTokenizer.from_pretrained(model_ckpt)

model_pegasus = AutoModelForSeq2SeqLM.from_pretrained(model_ckpt).to(device)
```

```
Downloading:   0%|          | 0.00/88.0 [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/1.12k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/1.91M [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/65.0 [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/2.28G [00:00<?, ?B/s]
```

In [6]:
```
def generate_batch_sized_chunks(list_of_elements, batch_size):
    """split the dataset into smaller batches that we can process simultaneously
    Yield successive batch-sized chunks from list_of_elements."""
    for i in range(0, len(list_of_elements), batch_size):
        yield list_of_elements[i : i + batch_size]
```

In [7]:
```
def calculate_metric_on_test_ds(dataset, metric, model, tokenizer,
                               batch_size=16, device=device,
                               column_text="article",
                                column_summary="highlights"):
    article_batches = list(generate_batch_sized_chunks(dataset[column_text], batch_size))
    target_batches = list(generate_batch_sized_chunks(dataset[column_summary], batch_size))

    for article_batch, target_batch in tqdm(
        zip(article_batches, target_batches), total=len(article_batches)):

        inputs = tokenizer(article_batch, max_length=1024,  truncation=True,
                        padding="max_length", return_tensors="pt")

        summaries = model.generate(input_ids=inputs["input_ids"].to(device),
                         attention_mask=inputs["attention_mask"].to(device),
                         length_penalty=0.8, num_beams=8, max_length=128)
        ''' parameter for length penalty ensures that the model does not generate sequences that are too long.

        # Finally, we decode the generated texts,
        # replace the  token, and add the decoded texts with the references to the metric.
        decoded_summaries = [tokenizer.decode(s, skip_special_tokens=True,
                                clean_up_tokenization_spaces=True)
               for s in summaries]

        decoded_summaries = [d.replace("", " ") for d in decoded_summaries]


        metric.add_batch(predictions=decoded_summaries, references=target_batch)

    #  Finally compute and return the ROUGE scores.
    score = metric.compute()
    return score
```

Load data

Link: https://huggingface.co/datasets/samsum

In [8]:
```
dataset_samsum = load_dataset("samsum")

split_lengths = [len(dataset_samsum[split])for split in dataset_samsum]

print(f"Split lengths: {split_lengths}")
print(f"Features: {dataset_samsum['train'].column_names}")
print("\nDialogue:")

print(dataset_samsum["test"][1]["dialogue"])

print("\nSummary:")

print(dataset_samsum["test"][1]["summary"])
```

```
Downloading builder script:   0%|          | 0.00/3.36k [00:00<?, ?B/s]
Downloading metadata:   0%|          | 0.00/1.58k [00:00<?, ?B/s]
Downloading readme:   0%|          | 0.00/6.88k [00:00<?, ?B/s]

Downloading and preparing dataset samsum/samsum to /root/.cache/huggingface/datasets/samsum/samsum/0.0.0/f1d7c6b7353e6de335d444e424dc002ef70d1277109

Downloading data:   0%|          | 0.00/2.94M [00:00<?, ?B/s]
Generating train split:   0%|          | 0/14732 [00:00<?, ? examples/s]
Generating test split:   0%|          | 0/819 [00:00<?, ? examples/s]
Generating validation split:   0%|          | 0/818 [00:00<?, ? examples/s]

Dataset samsum downloaded and prepared to /root/.cache/huggingface/datasets/samsum/samsum/0.0.0/f1d7c6b7353e6de335d444e424dc002ef70d1277109031327bc9

  0%|          | 0/3 [00:00<?, ?it/s]

Split lengths: [14732, 819, 818]
Features: ['id', 'dialogue', 'summary']

Dialogue:
Eric: MACHINE!
Rob: That's so gr8!
Eric: I know! And shows how Americans see Russian ;)
Rob: And it's really funny!
Eric: I know! I especially like the train part!
Rob: Hahaha! No one talks to the machine like that!
Eric: Is this his only stand-up?
Rob: Idk. I'll check.
Eric: Sure.
Rob: Turns out no! There are some of his stand-ups on youtube.
Eric: Gr8! I'll watch them now!
Rob: Me too!
Eric: MACHINE!
Rob: MACHINE!
Eric: TTYL?
Rob: Sure :)

Summary:
Eric and Rob are going to watch a stand-up on youtube.
```

Evaluating PEGASUS on SAMSum

In [9]:
```python
dataset_samsum['test'][0]['dialogue']
```

Out [9]:
```
"Hannah: Hey, do you have Betty's number?\nAmanda: Lemme check\nHannah: <file_gif>\nAmanda: Sorry, can't find it.\nAmanda: Ask
Larry\nAmanda: He called her last time we were at the park together\nHannah: I don't know him well\nHannah: <file_gif>\nAmanda: Don't be
shy, he's very nice\nHannah: If you say so..\nHannah: I'd rather you texted him\nAmanda: Just text him 🙂\nHannah: Urgh..
Alright\nHannah: Bye\nAmanda: Bye bye"
```

In [10]:
```python
pipe = pipeline('summarization', model = model_ckpt )

pipe_out = pipe(dataset_samsum['test'][0]['dialogue'] )

print(pipe_out)
```

```
Your max_length is set to 128, but you input_length is only 122. You might consider decreasing max_length manually, e.g.
summarizer('...', max_length=61)
```

```
[{'summary_text': "Amanda: Ask Larry Amanda: He called her last time we were at the park together .<n>Hannah: I'd rather you texted him .<n>Amanda:
```

In [11]:
```python
print(pipe_out[0]['summary_text'].replace(" .<n>", ".\n"))
```

```
Amanda: Ask Larry Amanda: He called her last time we were at the park together.
Hannah: I'd rather you texted him.
Amanda: Just text him .
```

In [12]:
```python
rouge_metric = load_metric('rouge')

score = calculate_metric_on_test_ds(dataset_samsum['test'], rouge_metric, model_pegasus, tokenizer, column_tex
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: 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: https://huggingface.co/docs/evaluate
  after removing the cwd from sys.path.
```

```
Downloading builder script:   0%|          | 0.00/2.16k [00:00<?, ?B/s]
```

```
  0%|          | 0/103 [00:00<?, ?it/s]▯[A
  1%|          | 1/103 [00:14<24:31, 14.42s/it]▯[A
  2%|▏         | 2/103 [00:24<19:39, 11.68s/it]▯[A
  3%|▎         | 3/103 [00:36<20:10, 12.10s/it]▯[A
  4%|▍         | 4/103 [00:49<20:34, 12.47s/it]▯[A
  5%|▍         | 5/103 [00:58<18:24, 11.27s/it]▯[A
  6%|▌         | 6/103 [01:12<19:15, 11.91s/it]▯[A
  7%|▋         | 7/103 [01:25<19:40, 12.29s/it]▯[A
  8%|▊         | 8/103 [01:37<19:28, 12.30s/it]▯[A
  9%|▊         | 9/103 [01:46<17:30, 11.18s/it]▯[A
 10%|▉         | 10/103 [01:59<18:23, 11.86s/it]▯[A
 11%|█         | 11/103 [02:09<17:09, 11.19s/it]▯[A
 12%|█▏        | 12/103 [02:22<17:55, 11.82s/it]▯[A
 13%|█▎        | 13/103 [02:34<17:46, 11.84s/it]▯[A
 14%|█▎        | 14/103 [02:47<17:55, 12.08s/it]▯[A
 15%|█▍        | 15/103 [03:00<18:14, 12.44s/it]▯[A
 16%|█▌        | 16/103 [03:13<18:26, 12.72s/it]▯[A
 17%|█▋        | 17/103 [03:26<18:14, 12.72s/it]▯[A
 17%|█▋        | 18/103 [03:35<16:38, 11.75s/it]▯[A
 18%|█▊        | 19/103 [03:46<16:00, 11.44s/it]▯[A
 19%|█▉        | 20/103 [03:59<16:35, 12.00s/it]▯[A
 20%|██        | 21/103 [04:13<16:54, 12.37s/it]▯[A
 21%|██▏       | 22/103 [04:24<16:11, 11.99s/it]▯[A
 22%|██▏       | 23/103 [04:33<15:01, 11.27s/it]▯[A
 23%|██▎       | 24/103 [04:47<15:38, 11.88s/it]▯[A
 24%|██▍       | 25/103 [04:58<15:21, 11.81s/it]▯[A
 25%|██▌       | 26/103 [05:11<15:33, 12.12s/it]▯[A
 26%|██▌       | 27/103 [05:25<15:49, 12.49s/it]▯[A
 27%|██▋       | 28/103 [05:36<15:02, 12.04s/it]▯[A
 28%|██▊       | 29/103 [05:45<13:57, 11.32s/it]▯[A
 29%|██▉       | 30/103 [05:59<14:31, 11.94s/it]▯[A
 30%|███       | 31/103 [06:11<14:25, 12.03s/it]▯[A
 31%|███       | 32/103 [06:22<13:59, 11.82s/it]▯[A
 32%|███▏      | 33/103 [06:35<14:19, 12.27s/it]▯[A
 33%|███▎      | 34/103 [06:46<13:24, 11.66s/it]▯[A
 34%|███▍      | 35/103 [06:54<11:59, 10.57s/it]▯[A
 35%|███▍      | 36/103 [07:06<12:25, 11.13s/it]▯[A
 36%|███▌      | 37/103 [07:19<12:57, 11.78s/it]▯[A
 37%|███▋      | 38/103 [07:33<13:17, 12.27s/it]▯[A
 38%|███▊      | 39/103 [07:46<13:25, 12.59s/it]▯[A
 39%|███▉      | 40/103 [07:59<13:12, 12.57s/it]▯[A
 40%|███▉      | 41/103 [08:08<12:06, 11.71s/it]▯[A
 41%|████      | 42/103 [08:19<11:34, 11.38s/it]▯[A
 42%|████▏     | 43/103 [08:32<11:58, 11.97s/it]▯[A
 43%|████▎     | 44/103 [08:40<10:22, 10.55s/it]▯[A
 44%|████▎     | 45/103 [08:53<10:59, 11.37s/it]▯[A
 45%|████▍     | 46/103 [09:02<10:03, 10.59s/it]▯[A
 46%|████▌     | 47/103 [09:14<10:29, 11.24s/it]▯[A
 47%|████▋     | 48/103 [09:23<09:33, 10.42s/it]▯[A
 48%|████▊     | 49/103 [09:32<09:06, 10.13s/it]▯[A
 49%|████▊     | 50/103 [09:46<09:48, 11.10s/it]▯[A
 50%|████▉     | 51/103 [09:56<09:16, 10.71s/it]▯[A
 50%|█████     | 52/103 [10:09<09:45, 11.47s/it]▯[A
 51%|█████▏    | 53/103 [10:21<09:52, 11.84s/it]▯[A
 52%|█████▏    | 54/103 [10:33<09:37, 11.78s/it]▯[A
 53%|█████▎    | 55/103 [10:46<09:47, 12.24s/it]▯[A
 54%|█████▍    | 56/103 [10:58<09:25, 12.02s/it]▯[A
 55%|█████▌    | 57/103 [11:10<09:10, 11.97s/it]▯[A
 56%|█████▋    | 58/103 [11:20<08:39, 11.55s/it]▯[A
 57%|█████▋    | 59/103 [11:34<08:50, 12.06s/it]▯[A
 58%|█████▊    | 60/103 [11:47<08:54, 12.42s/it]▯[A
 59%|█████▉    | 61/103 [11:57<08:12, 11.73s/it]▯[A
 60%|██████    | 62/103 [12:09<08:02, 11.77s/it]▯[A
 61%|██████    | 63/103 [12:21<07:52, 11.81s/it]▯[A
 62%|██████▏   | 64/103 [12:34<07:57, 12.25s/it]▯[A
 63%|██████▎   | 65/103 [12:47<07:55, 12.51s/it]▯[A
 64%|██████▍   | 66/103 [12:58<07:21, 11.93s/it]▯[A
 65%|██████▌   | 67/103 [13:08<06:53, 11.48s/it]▯[A
 66%|██████▌   | 68/103 [13:21<06:56, 11.91s/it]▯[A
 67%|██████▋   | 69/103 [13:30<06:17, 11.10s/it]▯[A
 68%|██████▊   | 70/103 [13:43<06:21, 11.57s/it]▯[A
 69%|██████▉   | 71/103 [13:55<06:12, 11.65s/it]▯[A
 70%|██████▉   | 72/103 [14:08<06:16, 12.15s/it]▯[A
 71%|███████   | 73/103 [14:17<05:38, 11.29s/it]▯[A
 72%|███████▏  | 74/103 [14:30<05:35, 11.58s/it]▯[A
 73%|███████▎  | 75/103 [14:40<05:12, 11.18s/it]▯[A
 74%|███████▍  | 76/103 [14:53<05:19, 11.83s/it]▯[A
 75%|███████▍  | 77/103 [15:06<05:18, 12.26s/it]▯[A
 76%|███████▌  | 78/103 [15:20<05:14, 12.57s/it]▯[A
 77%|███████▋  | 79/103 [15:33<05:07, 12.80s/it]▯[A
 78%|███████▊  | 80/103 [15:46<04:57, 12.95s/it]▯[A
 79%|███████▊  | 81/103 [15:59<04:45, 12.97s/it]▯[A
 80%|███████▉  | 82/103 [16:13<04:34, 13.06s/it]▯[A
 81%|████████  | 83/103 [16:22<03:56, 11.82s/it]▯[A
 82%|████████▏ | 84/103 [16:35<03:53, 12.27s/it]▯[A
 83%|████████▎ | 85/103 [16:48<03:46, 12.58s/it]▯[A
 83%|████████▎ | 86/103 [16:59<03:26, 12.14s/it]▯[A
 84%|████████▍ | 87/103 [17:13<03:19, 12.49s/it]▯[A
 85%|████████▌ | 88/103 [17:26<03:09, 12.61s/it]▯[A
 86%|████████▋ | 89/103 [17:39<02:59, 12.80s/it]▯[A
 87%|████████▋ | 90/103 [17:51<02:44, 12.68s/it]▯[A
 88%|████████▊ | 91/103 [18:03<02:28, 12.38s/it]▯[A
 89%|████████▉ | 92/103 [18:16<02:19, 12.66s/it]▯[A
 90%|█████████ | 93/103 [18:30<02:08, 12.85s/it]▯[A
 91%|█████████▏| 94/103 [18:42<01:53, 12.62s/it]▯[A
 92%|█████████▏| 95/103 [18:50<01:31, 11.45s/it]▯[A
 93%|█████████▎| 96/103 [19:04<01:23, 12.00s/it]▯[A
 94%|█████████▍| 97/103 [19:17<01:14, 12.38s/it]▯[A
 95%|█████████▌| 98/103 [19:30<01:03, 12.66s/it]▯[A
 96%|█████████▌| 99/103 [19:43<00:51, 12.85s/it]▯[A
 97%|█████████▋| 100/103 [19:56<00:38, 12.81s/it]▯[A
 98%|█████████▊| 101/103 [20:09<00:25, 12.74s/it]▯[A
 99%|█████████▉| 102/103 [20:22<00:12, 12.91s/it]▯[A
100%|██████████| 103/103 [20:25<00:00, 11.90s/it]
```

In [13]:
```python
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]
rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in rouge_names )

pd.DataFrame(rouge_dict, index = ['pegasus'])
```

Out [13]:

	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.015558	0.000295	0.015537	0.01554

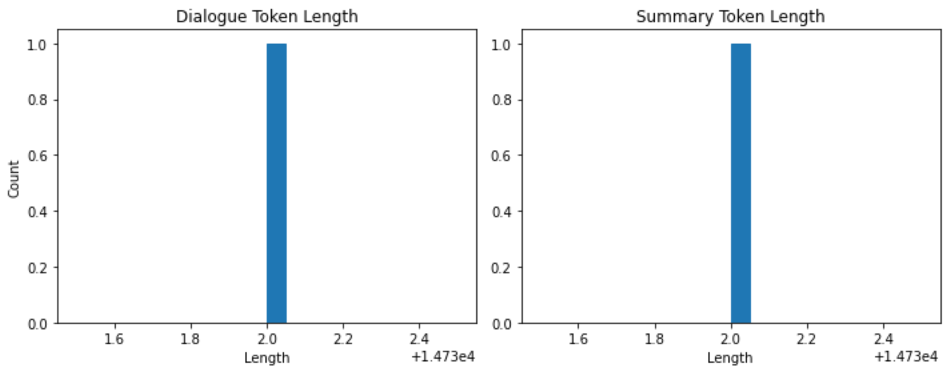
Histogram

In [ ]:
```python
dialogue_token_len = len([tokenizer.encode(s) for s in dataset_samsum['train']['dialogue']])

summary_token_len = len([tokenizer.encode(s) for s in dataset_samsum['train']['summary']])


fig, axes = plt.subplots(1, 2, figsize=(10, 4))
axes[0].hist(dialogue_token_len, bins = 20, color = 'C0', edgecolor = 'C0' )
axes[0].set_title("Dialogue Token Length")
axes[0].set_xlabel("Length")
axes[0].set_ylabel("Count")

axes[1].hist(summary_token_len, bins = 20, color = 'C0', edgecolor = 'C0' )
axes[1].set_title("Summary Token Length")
axes[1].set_xlabel("Length")
plt.tight_layout()
plt.show()
```



In [14]:
```python
def convert_examples_to_features(example_batch):
    input_encodings = tokenizer(example_batch['dialogue'] , max_length = 1024, truncation = True )
    with tokenizer.as_target_tokenizer():
        target_encodings = tokenizer(example_batch['summary'], max_length = 128, truncation = True )

    return {
        'input_ids' : input_encodings['input_ids'],
        'attention_mask': input_encodings['attention_mask'],
        'labels': target_encodings['input_ids']
    }

dataset_samsum_pt = dataset_samsum.map(convert_examples_to_features, batched = True)
```

```
  0%|          | 0/15 [00:00<?, ?ba/s]
```

```
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:3547: UserWarning: `as_target_tokenizer` is deprecated
and will be removed in v5 of Transformers. You can tokenize your labels by using the argument `text_target` of the regular `__call__`
method (either in the same call as your input texts if you use the same keyword arguments, or in a separate call.
  "`as_target_tokenizer` is deprecated and will be removed in v5 of Transformers. You can tokenize your "
```

```
  0%|          | 0/1 [00:00<?, ?ba/s]
  0%|          | 0/1 [00:00<?, ?ba/s]
```

In [15]:

```
from transformers import DataCollatorForSeq2Seq

seq2seq_data_collator = DataCollatorForSeq2Seq(tokenizer, model=model_pegasus)
```

In [17]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [18]:

```
%cd /content/drive/MyDrive/005_BOKTIAR_AHMED_BAPPY/My_classes/FSDS_NOV_10_AM
```

```
/content/drive/MyDrive/005_BOKTIAR_AHMED_BAPPY/My_classes/FSDS_NOV_10_AM
```

In [19]:

```
from transformers import TrainingArguments, Trainer

trainer_args = TrainingArguments(
    output_dir='pegasus-samsum', num_train_epochs=1, warmup_steps=500,
    per_device_train_batch_size=1, per_device_eval_batch_size=1,
    weight_decay=0.01, logging_steps=10,
    evaluation_strategy='steps', eval_steps=500, save_steps=1e6,
    gradient_accumulation_steps=16
)
```

In [20]:

```


trainer = Trainer(model=model_pegasus, args=trainer_args,
                  tokenizer=tokenizer, data_collator=seq2seq_data_collator,
                  train_dataset=dataset_samsum_pt["train"],
                  eval_dataset=dataset_samsum_pt["validation"])
```

In [21]:

```
trainer.train()
```

```
The following columns in the training set don't have a corresponding argument in `PegasusForConditionalGeneration.forward` and have
been ignored: summary, dialogue, id. If summary, dialogue, id are not expected by `PegasusForConditionalGeneration.forward`,  you can
safely ignore this message.
/usr/local/lib/python3.7/dist-packages/transformers/optimization.py:310: FutureWarning: This implementation of AdamW is deprecated and
will be removed in a future version. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warning=True` to
disable this warning
  FutureWarning,
***** Running training *****
  Num examples = 14732
  Num Epochs = 1
  Instantaneous batch size per device = 1
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 16
  Total optimization steps = 920
  Number of trainable parameters = 568699904
You're using a PegasusTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster than using
a method to encode the text followed by a call to the `pad` method to get a padded encoding.
```

[920/920 45:04, Epoch 0/1]

Step	Training Loss	Validation Loss
500	1.690200	1.488429

```
The following columns in the evaluation set don't have a corresponding argument in `PegasusForConditionalGeneration.forward` and have
been ignored: summary, dialogue, id. If summary, dialogue, id are not expected by `PegasusForConditionalGeneration.forward`,  you can
safely ignore this message.
***** Running Evaluation *****
  Num examples = 818
  Batch size = 1


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

Out [21]:

```
TrainOutput(global_step=920, training_loss=1.828826087454091, metrics={'train_runtime': 2707.8434, 'train_samples_per_second': 5.44,
'train_steps_per_second': 0.34, 'total_flos': 5526961323663360.0, 'train_loss': 1.828826087454091, 'epoch': 1.0})
```

In [30]:
```python
score = calculate_metric_on_test_ds(
    dataset_samsum['test'], rouge_metric, trainer.model, tokenizer, batch_size = 2, column_text = 'dialogue', 
)

rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in rouge_names )

pd.DataFrame(rouge_dict, index = [f'pegasus'] )
```

```
100%|██████████| 410/410 [13:08<00:00,  1.92s/it]
```

Out [30]:

	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.018618	0.000297	0.018493	0.018518

In [22]:
```python
## Save model
model_pegasus.save_pretrained("pegasus-samsum-model")
```

```
Configuration saved in pegasus-samsum-model/config.json
Model weights saved in pegasus-samsum-model/pytorch_model.bin
```

In [23]:
```python
## Save tokenizer
tokenizer.save_pretrained("tokenizer")
```

```
tokenizer config file saved in tokenizer/tokenizer_config.json
Special tokens file saved in tokenizer/special_tokens_map.json
```

Out [23]:
```
('tokenizer/tokenizer_config.json',
 'tokenizer/special_tokens_map.json',
 'tokenizer/spiece.model',
 'tokenizer/added_tokens.json',
 'tokenizer/tokenizer.json')
```

Test

In [24]:
```python
dataset_samsum = load_dataset("samsum")
```

```
WARNING:datasets.builder:Found cached dataset samsum
(/root/.cache/huggingface/datasets/samsum/samsum/0.0.0/f1d7c6b7353e6de335d444e424dc002ef70d1277109031327bc9cc6af5d3d46e)
```

```
  0%|          | 0/3 [00:00<?, ?it/s]
```

In [25]:
```python
tokenizer = AutoTokenizer.from_pretrained("tokenizer")
```

```
loading file spiece.model
loading file tokenizer.json
loading file added_tokens.json
loading file special_tokens_map.json
loading file tokenizer_config.json
```

In [27]:
```python
sample_text = dataset_samsum["test"][0]["dialogue"]

reference = dataset_samsum["test"][0]["summary"]
```

In [28]:
```python
gen_kwargs = {"length_penalty": 0.8, "num_beams":8, "max_length": 128}

pipe = pipeline("summarization", model="pegasus-samsum-model",tokenizer=tokenizer)
```

```
loading configuration file pegasus-samsum-model/config.json
Model config PegasusConfig {
  "_name_or_path": "pegasus-samsum-model",
  "activation_dropout": 0.1,
  "activation_function": "relu",
  "add_bias_logits": false,
  "add_final_layer_norm": true,
  "architectures": [
    "PegasusForConditionalGeneration"
  ],
  "attention_dropout": 0.1,
  "bos_token_id": 0,
  "classif_dropout": 0.0,
  "classifier_dropout": 0.0,
  "d_model": 1024,
  "decoder_attention_heads": 16,
  "decoder_ffn_dim": 4096,
  "decoder_layerdrop": 0.0,
  "decoder_layers": 16,
  "decoder_start_token_id": 0,
  "dropout": 0.1,
  "encoder_attention_heads": 16,
  "encoder_ffn_dim": 4096,
  "encoder_layerdrop": 0.0,
  "encoder_layers": 16,
  "eos_token_id": 1,
  "extra_pos_embeddings": 1,
  "forced_eos_token_id": 1,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2"
  },
  "init_std": 0.02,
  "is_encoder_decoder": true,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2
  },
  "length_penalty": 0.8,
  "max_length": 128,
  "max_position_embeddings": 1024,
  "min_length": 32,
  "model_type": "pegasus",
  "normalize_before": true,
  "normalize_embedding": false,
  "num_beams": 8,
  "num_hidden_layers": 16,
  "pad_token_id": 0,
  "scale_embedding": true,
  "static_position_embeddings": true,
  "torch_dtype": "float32",
  "transformers_version": "4.24.0",
  "use_cache": true,
  "vocab_size": 96103
}

loading configuration file pegasus-samsum-model/config.json
Model config PegasusConfig {
  "_name_or_path": "pegasus-samsum-model",
  "activation_dropout": 0.1,
  "activation_function": "relu",
  "add_bias_logits": false,
  "add_final_layer_norm": true,
  "architectures": [
    "PegasusForConditionalGeneration"
  ],
  "attention_dropout": 0.1,
  "bos_token_id": 0,
  "classif_dropout": 0.0,
  "classifier_dropout": 0.0,
  "d_model": 1024,
  "decoder_attention_heads": 16,
  "decoder_ffn_dim": 4096,
  "decoder_layerdrop": 0.0,
  "decoder_layers": 16,
  "decoder_start_token_id": 0,
  "dropout": 0.1,
  "encoder_attention_heads": 16,
  "encoder_ffn_dim": 4096,
  "encoder_layerdrop": 0.0,
  "encoder_layers": 16,
  "eos_token_id": 1,
  "extra_pos_embeddings": 1,
  "forced_eos_token_id": 1,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2"
  },
  "init_std": 0.02,
  "is_encoder_decoder": true,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2
  },
  "length_penalty": 0.8,
  "max_length": 128,
  "max_position_embeddings": 1024,
  "min_length": 32,
  "model_type": "pegasus",
  "normalize_before": true,
  "normalize_embedding": false,
  "num_beams": 8,
  "num_hidden_layers": 16,
  "pad_token_id": 0,
  "scale_embedding": true,
  "static_position_embeddings": true,
  "torch_dtype": "float32",
  "transformers_version": "4.24.0",
  "use_cache": true,
  "vocab_size": 96103
}

loading weights file pegasus-samsum-model/pytorch_model.bin
All model checkpoint weights were used when initializing PegasusForConditionalGeneration.

All the weights of PegasusForConditionalGeneration were initialized from the model checkpoint at pegasus-samsum-model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use PegasusForConditionalGeneration for
predictions without further training.
```

In [29]:

```python
print("Dialogue:")
print(sample_text)


print("\nReference Summary:")
print(reference)


print("\nModel Summary:")
print(pipe(sample_text, **gen_kwargs)[0]["summary_text"])
```

```
Your max_length is set to 128, but you input_length is only 122. You might consider decreasing max_length manually, e.g.
summarizer('...', max_length=61)
```

```
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

Reference Summary:
Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry.

Model Summary:
Amanda can't find Betty's number. Larry called Betty last time they were at the park together. Hannah wants Amanda to text Larry. Amanda will text L
```