

# Generative AI Use Case: Summarize Dialogue

Welcome to the practical side of this course. In this lab you will do the dialogue summarization task using generative AI. You will explore how the input text affects the output of the model, and perform prompt engineering to direct it towards the task you need. By comparing zero shot, one shot, and few shot inferences, you will take the first step towards prompt engineering and see how it can enhance the generative output of Large Language Models.

## Table of Contents

- 1 - Set up Kernel and Required Dependencies
- 2 - Summarize Dialogue without Prompt Engineering
- 3 - Summarize Dialogue with an Instruction Prompt
  - 3.1 - Zero Shot Inference with an Instruction Prompt
  - 3.2 - Zero Shot Inference with the Prompt Template from FLAN-T5
- 4 - Summarize Dialogue with One Shot and Few Shot Inference
  - 4.1 - One Shot Inference
  - 4.2 - Few Shot Inference
- 5 - Generative Configuration Parameters for Inference

## 1 - Set up Kernel and Required Dependencies

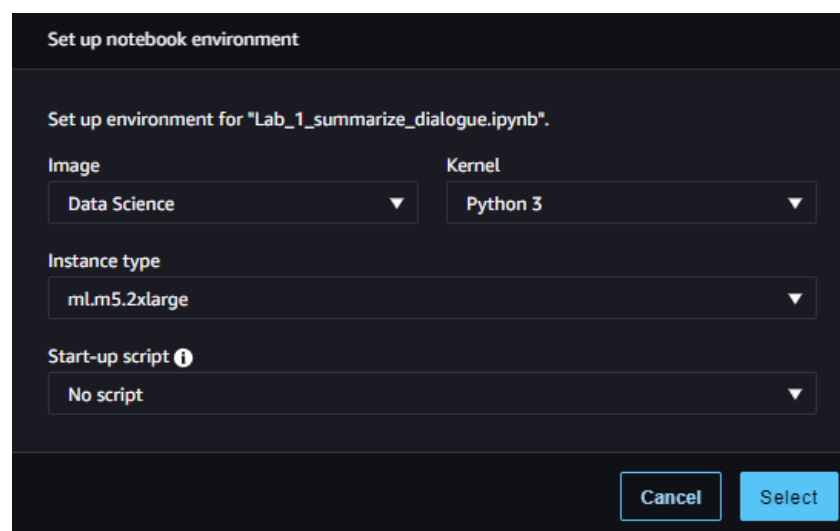
This notebook has been modified to run locally with Jupyter Notebook server and VScode

Config for running in AWS

First, check that the correct kernel is chosen.

Data Science Python 3 8 vCPU + 32 GiB

You can click on that (top right of the screen) to see and check the details of the image, kernel, and instance type.



Now install the required packages to use PyTorch and Hugging Face transformers and datasets.



The next cell may take a few minutes to run. Please be patient.  
Ignore the warnings and errors, along with the note about restarting the kernel at the end.

```
In [ ]: # Did comment this instalaiton due to version issues for notebook Local run.
```

```
%%pip install --upgrade pip
%%pip install --disable-pip-version-check \
```

```
# torch==1.13.1 \
# torchdata==0.5.1 --quiet

##pip install \
# transformers==4.27.2 \ # better to install latest for local notebook run
# datasets==2.11.0 --quiet # did have errors with this version. Had to update to latest 2.16
```

```
Requirement already satisfied: pip in c:\miniconda3\lib\site-packages (23.3.1)
Collecting pip
  Downloading pip-23.3.2-py3-none-any.whl.metadata (3.5 kB)
Downloading pip-23.3.2-py3-none-any.whl (2.1 MB)
----- 0.0/2.1 MB ? eta -:--:--
----- 0.3/2.1 MB 8.6 MB/s eta 0:00:01
----- 0.9/2.1 MB 14.7 MB/s eta 0:00:01
----- 2.1/2.1 MB 19.1 MB/s eta 0:00:01
----- 2.1/2.1 MB 16.8 MB/s eta 0:00:00

Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.3.1
    Uninstalling pip-23.3.1:
      Successfully uninstalled pip-23.3.1
Successfully installed pip-23.3.2
Note: you may need to restart the kernel to use updated packages.
Note: you may need to restart the kernel to use updated packages.
Note: you may need to restart the kernel to use updated packages.
```

Load the datasets, Large Language Model (LLM), tokenizer, and configurator. Do not worry if you do not understand yet all of those components - they will be described and discussed later in the notebook.

```
In [ ]: from datasets import load_dataset
        from transformers import AutoModelForSeq2SeqLM
        from transformers import AutoTokenizer
        from transformers import GenerationConfig
```

## 2 - Summarize Dialogue without Prompt Engineering

In this use case, you will be generating a summary of a dialogue with the pre-trained Large Language Model (LLM) FLAN-T5 from Hugging Face. The list of available models in the Hugging Face `transformers` package can be found [here](#).

Let's upload some simple dialogues from the [DialogSum](#) Hugging Face dataset. This dataset contains 10,000+ dialogues with the corresponding manually labeled summaries and topics.

```
In [ ]: huggingface_dataset_name = "knkarthick/dialogsum"

        dataset = load_dataset(huggingface_dataset_name)
```

```
In [ ]: # Lets see the dataset structure
        dataset
```

```
Out[ ]: DatasetDict({
  train: Dataset({
    features: ['id', 'dialogue', 'summary', 'topic'],
    num_rows: 12460
  })
  validation: Dataset({
    features: ['id', 'dialogue', 'summary', 'topic'],
    num_rows: 500
  })
  test: Dataset({
    features: ['id', 'dialogue', 'summary', 'topic'],
    num_rows: 1500
  })
})
```

Print a couple of dialogues with their baseline summaries.

```
In [ ]: example_indices = [40, 200]

        dash_line = '-'.join(' ' for x in range(100))

        for i, index in enumerate(example_indices):
            print(dash_line)
            print('Example ', i + 1)
            print(dash_line)
            print('INPUT DIALOGUE:')
            print(dataset['test'][index]['dialogue']) # <-----
            print(dash_line)
```

```

print('BASELINE HUMAN SUMMARY:')
print(dataset['test'][index]['summary']) # <-----
print(dash_line)
print()

```

-----  
Example 1  
-----

INPUT DIALOGUE:

```

#Person1#: What time is it, Tom?
#Person2#: Just a minute. It's ten to nine by my watch.
#Person1#: Is it? I had no idea it was so late. I must be off now.
#Person2#: What's the hurry?
#Person1#: I must catch the nine-thirty train.
#Person2#: You've plenty of time yet. The railway station is very close. It won't take more than twenty minutes to get there.

```

-----  
BASELINE HUMAN SUMMARY:

```

#Person1# is in a hurry to catch a train. Tom tells #Person1# there is plenty of time.

```

-----

-----  
Example 2  
-----

INPUT DIALOGUE:

```

#Person1#: Have you considered upgrading your system?
#Person2#: Yes, but I'm not sure what exactly I would need.
#Person1#: You could consider adding a painting program to your software. It would allow you to make up your own flyers and banners for advertising.
#Person2#: That would be a definite bonus.
#Person1#: You might also want to upgrade your hardware because it is pretty outdated now.
#Person2#: How can we do that?
#Person1#: You'd probably need a faster processor, to begin with. And you also need a more powerful hard disc, more memory and a faster modem. Do you have a CD-ROM drive?
#Person2#: No.
#Person1#: Then you might want to add a CD-ROM drive too, because most new software programs are coming out on Cds.
#Person2#: That sounds great. Thanks.

```

-----  
BASELINE HUMAN SUMMARY:

```

#Person1# teaches #Person2# how to upgrade software and hardware in #Person2#'s system.

```

-----

Load the [FLAN-T5 model](#), creating an instance of the `AutoModelForSeq2SeqLM` class with the `.from_pretrained()` method.

```
In [ ]: model_name='google/flan-t5-base'
```

```
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
```

```

pytorch_model.bin:  0%|          | 0.00/990M [00:00<?, ?B/s]
c:\Miniconda3\lib\site-packages\huggingface_hub\file_download.py:149: UserWarning: `huggingface_hub` cache-system uses symlinks by default to efficiently store duplicated files but your machine does not support them in C:\Users\StaFaka\.cache\huggingface\hub\models--google--flan-t5-base. Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details, see https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to see activate developer mode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
  warnings.warn(message)
generation_config.json:  0%|          | 0.00/147 [00:00<?, ?B/s]

```

To perform encoding and decoding, you need to work with text in a tokenized form. **Tokenization** is the process of splitting texts into smaller units that can be processed by the LLM models.

Download the tokenizer for the FLAN-T5 model using `AutoTokenizer.from_pretrained()` method. Parameter `use_fast` switches on fast tokenizer. At this stage, there is no need to go into the details of that, but you can find the tokenizer parameters in the [documentation](#).

```
In [ ]: tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=True) # Load tokenizer for our model (model_name='google/f
```

```

tokenizer_config.json:  0%|          | 0.00/2.54k [00:00<?, ?B/s]
spiece.model:  0%|          | 0.00/792k [00:00<?, ?B/s]
tokenizer.json:  0%|          | 0.00/2.42M [00:00<?, ?B/s]
special_tokens_map.json:  0%|          | 0.00/2.20k [00:00<?, ?B/s]

```

Test the tokenizer encoding and decoding a simple sentence:

```
In [ ]: sentence = "What time is it, Tom?"
```

```
sentence_encoded = tokenizer(sentence, return_tensors='pt')
```

```

sentence_decoded = tokenizer.decode(
    sentence_encoded["input_ids"][0], # get the first sentence from tensor in List
    skip_special_tokens=True
)

```

```

print('ENCODED SENTENCE:')
print(sentence_encoded["input_ids"][0])
print('\nDECODED SENTENCE:')
print(sentence_decoded)

```

```

ENCODED SENTENCE:
tensor([ 363,  97,  19,  34,   6, 3059,  58,   1])

```

```

DECODED SENTENCE:
What time is it, Tom?

```

```

In [ ]: # Lets see structure of the tokenizer variable
        sentence_encoded

```

```

Out[ ]: {'input_ids': tensor([[ 363,  97,  19,  34,   6, 3059,  58,   1]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1]])}

```

Now it's time to explore how well the base LLM summarizes a dialogue without any prompt engineering. **Prompt engineering** is an act of a human changing the **prompt** (input) to improve the response for a given task.

```

In [ ]: for i, index in enumerate(example_indices):
        dialogue = dataset['test'][index]['dialogue']
        summary = dataset['test'][index]['summary']

        inputs = tokenizer(dialogue, return_tensors='pt')
        output = tokenizer.decode(
            model.generate(
                inputs["input_ids"],
                max_new_tokens=50,
            )[0],
            skip_special_tokens=True
        )

        print(dash_line)
        print('Example ', i + 1)
        print(dash_line)
        print(f'INPUT PROMPT:\n{dialogue}')
        print(dash_line)
        print(f'BASELINE HUMAN SUMMARY:\n{summary}')
        print(dash_line)
        print(f'MODEL GENERATION - WITHOUT PROMPT ENGINEERING:\n{output}\n')

```

-----  
Example 1  
-----

INPUT PROMPT:

```

#Person1#: What time is it, Tom?
#Person2#: Just a minute. It's ten to nine by my watch.
#Person1#: Is it? I had no idea it was so late. I must be off now.
#Person2#: What's the hurry?
#Person1#: I must catch the nine-thirty train.
#Person2#: You've plenty of time yet. The railway station is very close. It won't take more than twenty minutes to get there.

```

-----  
BASELINE HUMAN SUMMARY:

```

#Person1# is in a hurry to catch a train. Tom tells #Person1# there is plenty of time.

```

-----  
MODEL GENERATION - WITHOUT PROMPT ENGINEERING:

```

Person1: It's ten to nine.

```

-----  
Example 2  
-----

INPUT PROMPT:

```

#Person1#: Have you considered upgrading your system?
#Person2#: Yes, but I'm not sure what exactly I would need.
#Person1#: You could consider adding a painting program to your software. It would allow you to make up your own flyers and banners for advertising.
#Person2#: That would be a definite bonus.
#Person1#: You might also want to upgrade your hardware because it is pretty outdated now.
#Person2#: How can we do that?
#Person1#: You'd probably need a faster processor, to begin with. And you also need a more powerful hard disc, more memory and a faster modem. Do you have a CD-ROM drive?
#Person2#: No.
#Person1#: Then you might want to add a CD-ROM drive too, because most new software programs are coming out on Cds.
#Person2#: That sounds great. Thanks.

```

-----  
BASELINE HUMAN SUMMARY:

```

#Person1# teaches #Person2# how to upgrade software and hardware in #Person2#'s system.

```

-----  
MODEL GENERATION - WITHOUT PROMPT ENGINEERING:

```

#Person1#: I'm thinking of upgrading my computer.

```

You can see that the guesses of the model make some sense, but it doesn't seem to be sure what task it is supposed to accomplish.

Seems it just makes up the next sentence in the dialogue. Prompt engineering can help here.

## 3 - Summarize Dialogue with an Instruction Prompt

Prompt engineering is an important concept in using foundation models for text generation. You can check out [this blog](#) from Amazon Science for a quick introduction to prompt engineering.

### 3.1 - Zero Shot Inference with an Instruction Prompt

In order to instruct the model to perform a task - summarize a dialogue - you can take the dialogue and convert it into an instruction prompt. This is often called **zero shot inference**. You can check out [this blog from AWS](#) for a quick description of what zero shot learning is and why it is an important concept to the LLM model.

Wrap the dialogue in a descriptive instruction and see how the generated text will change:

```
In [ ]: for i, index in enumerate(example_indices):
        dialogue = dataset['test'][index]['dialogue']
        summary = dataset['test'][index]['summary']

        prompt = f"""
        How many people are involved in conversation? and what are their names.

        {dialogue}

        Summary:
        """

        # Input constructed prompt instead of the dialogue.
        inputs = tokenizer(prompt, return_tensors='pt')
        output = tokenizer.decode(
            model.generate(
                inputs["input_ids"],
                max_new_tokens=50,
            )[0],
            skip_special_tokens=True
        )

        print(dash_line)
        print('Example ', i + 1)
        print(dash_line)
        print(f'INPUT PROMPT:\n{prompt}')
        print(dash_line)
        print(f'BASELINE HUMAN SUMMARY:\n{summary}')
        print(dash_line)
        print(f'MODEL GENERATION - ZERO SHOT:\n{output}\n')
```

-----  
Example 1  
-----

INPUT PROMPT:

How many people are involved in conversation? and what are their names.

#Person1#: What time is it, Tom?  
#Person2#: Just a minute. It's ten to nine by my watch.  
#Person1#: Is it? I had no idea it was so late. I must be off now.  
#Person2#: What's the hurry?  
#Person1#: I must catch the nine-thirty train.  
#Person2#: You've plenty of time yet. The railway station is very close. It won't take more than twenty minutes to get there.

Summary:

-----  
BASELINE HUMAN SUMMARY:

#Person1# is in a hurry to catch a train. Tom tells #Person1# there is plenty of time.

-----  
MODEL GENERATION - ZERO SHOT:

The conversation is about Tom and his train.  
-----

-----  
Example 2  
-----

INPUT PROMPT:

How many people are involved in conversation? and what are their names.

#Person1#: Have you considered upgrading your system?  
#Person2#: Yes, but I'm not sure what exactly I would need.  
#Person1#: You could consider adding a painting program to your software. It would allow you to make up your own flyers and banners for advertising.  
#Person2#: That would be a definite bonus.  
#Person1#: You might also want to upgrade your hardware because it is pretty outdated now.  
#Person2#: How can we do that?  
#Person1#: You'd probably need a faster processor, to begin with. And you also need a more powerful hard disc, more memory and a faster modem. Do you have a CD-ROM drive?  
#Person2#: No.  
#Person1#: Then you might want to add a CD-ROM drive too, because most new software programs are coming out on Cds.  
#Person2#: That sounds great. Thanks.

Summary:

-----  
BASELINE HUMAN SUMMARY:

#Person1# teaches #Person2# how to upgrade software and hardware in #Person2#'s system.

-----  
MODEL GENERATION - ZERO SHOT:

#Person1: Have you considered upgrading your system? #Person2: Yes, but I'm not sure what exactly I would need. #Person1: You could consider adding a painting program to your software. #P

This is much better! But the model still does not pick up on the nuance of the conversations though.

#### Exercise:

- Experiment with the `prompt` text and see how the inferences will be changed. Will the inferences change if you end the prompt with just empty string vs. `Summary: ?`
- Try to rephrase the beginning of the `prompt` text from `Summarize the following conversation.` to something different - and see how it will influence the generated output.

## 3.2 - Zero Shot Inference with the Prompt Template from FLAN-T5

Let's use a slightly different prompt. FLAN-T5 has many prompt templates that are published for certain tasks [here](#). In the following code, you will use one of the [pre-built FLAN-T5 prompts](#):

```
In [ ]: for i, index in enumerate(example_indices):
        dialogue = dataset['test'][index]['dialogue']
        summary = dataset['test'][index]['summary']

        prompt = f"""
Dialogue:

{dialogue}

What was going on?
"""

        inputs = tokenizer(prompt, return_tensors='pt')
```

```

output = tokenizer.decode(
    model.generate(
        inputs["input_ids"],
        max_new_tokens=50,
    )[0],
    skip_special_tokens=True
)

print(dash_line)
print('Example ', i + 1)
print(dash_line)
print(f'INPUT PROMPT:\n{prompt}')
print(dash_line)
print(f'BASELINE HUMAN SUMMARY:\n{summary}\n')
print(dash_line)
print(f'MODEL GENERATION - ZERO SHOT:\n{output}\n') # <----- model output

```

-----  
Example 1  
-----

INPUT PROMPT:

Dialogue:

```

#Person1#: What time is it, Tom?
#Person2#: Just a minute. It's ten to nine by my watch.
#Person1#: Is it? I had no idea it was so late. I must be off now.
#Person2#: What's the hurry?
#Person1#: I must catch the nine-thirty train.
#Person2#: You've plenty of time yet. The railway station is very close. It won't take more than twenty minutes to get there.

```

What was going on?

-----  
BASELINE HUMAN SUMMARY:

#Person1# is in a hurry to catch a train. Tom tells #Person1# there is plenty of time.

-----  
MODEL GENERATION - ZERO SHOT:

Tom is late for the train.  
-----

-----  
Example 2  
-----

INPUT PROMPT:

Dialogue:

```

#Person1#: Have you considered upgrading your system?
#Person2#: Yes, but I'm not sure what exactly I would need.
#Person1#: You could consider adding a painting program to your software. It would allow you to make up your own flyers and banners for advertising.
#Person2#: That would be a definite bonus.
#Person1#: You might also want to upgrade your hardware because it is pretty outdated now.
#Person2#: How can we do that?
#Person1#: You'd probably need a faster processor, to begin with. And you also need a more powerful hard disc, more memory and a faster modem. Do you have a CD-ROM drive?
#Person2#: No.
#Person1#: Then you might want to add a CD-ROM drive too, because most new software programs are coming out on Cds.
#Person2#: That sounds great. Thanks.

```

What was going on?

-----  
BASELINE HUMAN SUMMARY:

#Person1# teaches #Person2# how to upgrade software and hardware in #Person2#'s system.  
-----

MODEL GENERATION - ZERO SHOT:

#Person1#: You could add a painting program to your software. #Person2#: That would be a bonus. #Person1#: You might also want to upgrade your hardware. #Person1#

Notice that this prompt from FLAN-T5 did help a bit, but still struggles to pick up on the nuance of the conversation. This is what you will try to solve with the few shot inferencing.

## 4 - Summarize Dialogue with One Shot and Few Shot Inference

**One shot and few shot inference** are the practices of providing an LLM with either one or more full examples of prompt-response pairs that match your task - before your actual prompt that you want completed. This is called "in-context learning" and puts your model into a state that understands your specific task. You can read more about it in [this blog from HuggingFace](#).

## 4.1 - One Shot Inference

Let's build a function that takes a list of `example_indices_full`, generates a prompt with full examples, then at the end appends the prompt which you want the model to complete (`example_index_to_summarize`). You will use the same FLAN-T5 prompt template from section 3.2.

```
In [ ]: def make_prompt(example_indices_full, example_index_to_summarize):
        prompt = ''
        for index in example_indices_full:
            dialogue = dataset['test'][index]['dialogue']
            summary = dataset['test'][index]['summary']

            # The stop sequence '{summary}\n\n' is important for FLAN-T5. Other models may have their own preferred stop seq
            prompt += f"""
Dialogue:

{dialogue}

What was going on?
{summary}

"""

            dialogue = dataset['test'][example_index_to_summarize]['dialogue']

            prompt += f"""
Dialogue:

{dialogue}

What was going on?
"""

        return prompt
```

Construct the prompt to perform one shot inference:

```
In [ ]: example_indices_full = [40]
        example_index_to_summarize = 200

        one_shot_prompt = make_prompt(example_indices_full, example_index_to_summarize)

        print(one_shot_prompt)
```

Dialogue:

```
#Person1#: What time is it, Tom?
#Person2#: Just a minute. It's ten to nine by my watch.
#Person1#: Is it? I had no idea it was so late. I must be off now.
#Person2#: What's the hurry?
#Person1#: I must catch the nine-thirty train.
#Person2#: You've plenty of time yet. The railway station is very close. It won't take more than twenty minutes to get there.
```

What was going on?

#Person1# is in a hurry to catch a train. Tom tells #Person1# there is plenty of time.

Dialogue:

```
#Person1#: Have you considered upgrading your system?
#Person2#: Yes, but I'm not sure what exactly I would need.
#Person1#: You could consider adding a painting program to your software. It would allow you to make up your own flyers and banners for advertising.
#Person2#: That would be a definite bonus.
#Person1#: You might also want to upgrade your hardware because it is pretty outdated now.
#Person2#: How can we do that?
#Person1#: You'd probably need a faster processor, to begin with. And you also need a more powerful hard disc, more memory and a faster modem. Do you have a CD-ROM drive?
#Person2#: No.
#Person1#: Then you might want to add a CD-ROM drive too, because most new software programs are coming out on Cds.
#Person2#: That sounds great. Thanks.
```

What was going on?

Now pass this prompt to perform the one shot inference:

```
In [ ]: summary = dataset['test'][example_index_to_summarize]['summary']
```



```

inputs = tokenizer(one_shot_prompt, return_tensors='pt')
output = tokenizer.decode(
    model.generate(
        inputs["input_ids"],
        max_new_tokens=50,
    )[0],
    skip_special_tokens=True
)

```

```

print(dash_line)
print(f'BASELINE HUMAN SUMMARY:\n{summary}\n')
print(dash_line)
print(f'MODEL GENERATION - ONE SHOT:\n{output}')

```

-----

BASELINE HUMAN SUMMARY:

#Person1# teaches #Person2# how to upgrade software and hardware in #Person2#'s system.

-----

MODEL GENERATION - ONE SHOT:

#Person1 wants to upgrade his system. #Person2 wants to add a painting program to his software. #Person1 wants to add a CD-ROM drive.

## 4.2 - Few Shot Inference

Let's explore few shot inference by adding two more full dialogue-summary pairs to your prompt.

```

In [ ]: example_indices_full = [42, 81, 121]
        example_index_to_summarize = 200

few_shot_prompt = make_prompt(example_indices_full, example_index_to_summarize)

print(few_shot_prompt)

```

Dialogue:

#Person1#: I don't know how to adjust my life. Would you give me a piece of advice?  
#Person2#: You look a bit pale, don't you?  
#Person1#: Yes, I can't sleep well every night.  
#Person2#: You should get plenty of sleep.  
#Person1#: I drink a lot of wine.  
#Person2#: If I were you, I wouldn't drink too much.  
#Person1#: I often feel so tired.  
#Person2#: You better do some exercise every morning.  
#Person1#: I sometimes find the shadow of death in front of me.  
#Person2#: Why do you worry about your future? You're very young, and you'll make great contribution to the world. I hope you take my advice.

What was going on?

#Person1# wants to adjust #Person1#'s life and #Person2# suggests #Person1# be positive and stay healthy.

Dialogue:

#Person1#: Hello, are you Muriel Douglas?  
#Person2#: Yes, and you must be James. It's nice to meet you at long last.  
#Person1#: Yes, you too. Thanks for agreeing to meet with us about the new account. My associate, Susan Kim, should be here any minute. Would you like something to drink while we're waiting?  
#Person2#: No, thanks. I'm fine. Did you have a nice holiday?  
#Person1#: Yes, I did. My family and I went to Tahoe to ski and the weather was great. How about you?  
#Person2#: I stayed in L. A. and it was sunny the entire weekend. We spent most of the time at home but we did go see King Kong on Christmas day.  
#Person1#: How did you like it?  
#Person2#: It was better than I expected. But, you know, I think I would have enjoyed skiing in Tahoe even better. Do you go there often?  
#Person1#: No, not much. My wife doesn't like to ski. She prefers vacationing where it's warmer, like Hawaii.  
#Person2#: I don't blame her. I really enjoyed it there when we went a few years ago. I'd like to go back sometime soon.  
#Person1#: Yes, me too. Oh, here's Susan now. Let me introduce you.

What was going on?

Muriel Douglas and James meet each other and talk about what they have done during the holiday.

Dialogue:

#Person1#: Hello, I bought the pendant in your shop, just before.  
#Person2#: Yes. Thank you very much.  
#Person1#: Now I come back to the hotel and try to show it to my friend, the pendant is broken, I'm afraid.  
#Person2#: Oh, is it?  
#Person1#: Would you change it to a new one?  
#Person2#: Yes, certainly. You have the receipt?  
#Person1#: Yes, I do.  
#Person2#: Then would you kindly come to our shop with the receipt by 10 o'clock? We will replace it.  
#Person1#: Thank you so much.

What was going on?

#Person1# goes back to #Person2#'s shop to replace a broken pendant.

Dialogue:

#Person1#: Have you considered upgrading your system?  
#Person2#: Yes, but I'm not sure what exactly I would need.  
#Person1#: You could consider adding a painting program to your software. It would allow you to make up your own flyers and banners for advertising.  
#Person2#: That would be a definite bonus.  
#Person1#: You might also want to upgrade your hardware because it is pretty outdated now.  
#Person2#: How can we do that?  
#Person1#: You'd probably need a faster processor, to begin with. And you also need a more powerful hard disc, more memory and a faster modem. Do you have a CD-ROM drive?  
#Person2#: No.  
#Person1#: Then you might want to add a CD-ROM drive too, because most new software programs are coming out on Cds.  
#Person2#: That sounds great. Thanks.

What was going on?

Now pass this prompt to perform a few shot inference:

```
In [ ]: summary = dataset['test'][example_index_to_summarize]['summary']

inputs = tokenizer(few_shot_prompt, return_tensors='pt')
output = tokenizer.decode(
    model.generate(
        inputs["input_ids"],
        max_new_tokens=50,
    )[0],
```

```

        skip_special_tokens=True
    )

    print(dash_line)
    print(f'BASELINE HUMAN SUMMARY:\n{summary}\n')
    print(dash_line)
    print(f'MODEL GENERATION - FEW SHOT:\n{output}')

```

Token indices sequence length is longer than the specified maximum sequence length for this model (819 > 512). Running this sequence through the model will result in indexing errors

-----

BASELINE HUMAN SUMMARY:

#Person1# teaches #Person2# how to upgrade software and hardware in #Person2#'s system.

-----

MODEL GENERATION - FEW SHOT:

#Person1 wants to upgrade his system. #Person2 wants to add a painting program to his software. #Person1 wants to upgrade his hardware.

In this case, few shot did not provide much of an improvement over one shot inference. And, anything above 5 or 6 shot will typically not help much, either. Also, you need to make sure that you do not exceed the model's input-context length which, in our case, is 512 tokens. Anything above the context length will be ignored.

However, you can see that feeding in at least one full example (one shot) provides the model with more information and qualitatively improves the summary overall.

#### Exercise:

Experiment with the few shot inferencing.

- Choose different dialogues - change the indices in the `example_indices_full` list and `example_index_to_summarize` value.
- Change the number of shots. Be sure to stay within the model's 512 context length, however.

How well does few shot inferencing work with other examples?

## 5 - Generative Configuration Parameters for Inference

You can change the configuration parameters of the `generate()` method to see a different output from the LLM. So far the only parameter that you have been setting was `max_new_tokens=50`, which defines the maximum number of tokens to generate. A full list of available parameters can be found in the [Hugging Face Generation documentation](#).

A convenient way of organizing the configuration parameters is to use `GenerationConfig` class.

#### Exercise:

Change the configuration parameters to investigate their influence on the output.

Putting the parameter `do_sample = True`, you activate various decoding strategies which influence the next token from the probability distribution over the entire vocabulary. You can then adjust the outputs changing `temperature` and other parameters (such as `top_k` and `top_p`).

Uncomment the lines in the cell below and rerun the code. Try to analyze the results. You can read some comments below.

```

In [ ]: # generation_config = GenerationConfig(max_new_tokens=50)
        # generation_config = GenerationConfig(max_new_tokens=10)
        generation_config = GenerationConfig(max_new_tokens=50, do_sample=True, temperature=0.1)
        # generation_config = GenerationConfig(max_new_tokens=50, do_sample=True, temperature=0.5)
        # generation_config = GenerationConfig(max_new_tokens=50, do_sample=True, temperature=1.0)

        inputs = tokenizer(few_shot_prompt, return_tensors='pt')
        output = tokenizer.decode(
            model.generate(
                inputs["input_ids"],
                generation_config=generation_config,
            )[0],
            skip_special_tokens=True
        )

        print(dash_line)
        print(f'MODEL GENERATION - FEW SHOT:\n{output}')
        print(dash_line)
        print(f'BASELINE HUMAN SUMMARY:\n{summary}\n')

```

```
-----  
MODEL GENERATION - FEW SHOT:  
#Person1 wants to upgrade his system and hardware.  
-----
```

```
BASELINE HUMAN SUMMARY:  
#Person1# teaches #Person2# how to upgrade software and hardware in #Person2#'s system.
```

Comments related to the choice of the parameters in the code cell above:

- Choosing `max_new_tokens=10` will make the output text too short, so the dialogue summary will be cut.
- Putting `do_sample = True` and changing the temperature value you get more flexibility in the output.

As you can see, prompt engineering can take you a long way for this use case, but there are some limitations. Next, you will start to explore how you can use fine-tuning to help your LLM to understand a particular use case in better depth!

In [ ]: