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

First, check that the correct kernel is chosen.

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

> Please make sure that you choose **ml.m5.2xlarge** instance type.
> To find that instance type, you might have to scroll down to the "All Instances" section in the dropdown.
> Choice of another instance type might cause training failure/kernel halt/account deactivation.

In [2]:
```
import os

instance_type_expected = 'ml-m5-2xlarge'
instance_type_current = os.environ.get('HOSTNAME')

print(f'Expected instance type: instance-datascience-{instance_type_expected}')
print(f'Currently chosen instance type: {instance_type_current}')

assert instance_type_expected in instance_type_current, f'ERROR. You selected the {instance_type_current} instance type. Please select {instance_type_expected} inste
print("Instance type has been chosen correctly.")
```

```
Expected instance type: instance-datascience-ml-m5-2xlarge
Currently chosen instance type: instance-datascience-ml-m5-2xlarge
Instance type has been chosen correctly.
```

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 [3]:
```
%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 \
    datasets==2.11.0  --quiet
```

```
Requirement already satisfied: pip in /opt/conda/lib/python3.10/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)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 17.2 MB/s eta 0:00:0000:01
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
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtu
al environment instead: https://pip.pypa.io/warnings/venv
Note: you may need to restart the kernel to use updated packages.
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtu
al environment instead: https://pip.pypa.io/warnings/venv
Note: you may need to restart the kernel to use updated packages.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency
conflicts.
spyder 5.3.3 requires pyqt5<5.16, which is not installed.
spyder 5.3.3 requires pyqtwebengine<5.16, which is not installed.
pathos 0.3.1 requires dill>=0.3.7, but you have dill 0.3.6 which is incompatible.
pathos 0.3.1 requires multiprocess>=0.70.15, but you have multiprocess 0.70.14 which is incompatible.
sagemaker 2.199.0 requires urllib3<1.27, but you have urllib3 2.1.0 which is incompatible.
spyder 5.3.3 requires ipython<8.0.0,>=7.31.1, but you have ipython 8.18.1 which is incompatible.
spyder 5.3.3 requires pylint<3.0,>=2.5.0, but you have pylint 3.0.2 which is incompatible.
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtu
al environment instead: https://pip.pypa.io/warnings/venv
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 [4]:
```
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 [5]:
```
huggingface_dataset_name = "knkarthick/dialogsum"

dataset = load_dataset(huggingface_dataset_name) # Loading DialogSum Hugging Face Dataset
```

```
Downloading readme:   0%|          | 0.00/4.65k [00:00<?, ?B/s]
Downloading and preparing dataset csv/knkarthick--dialogsum to /root/.cache/huggingface/datasets/knkarthick___csv/knkarthick--dialogsum-cd36827d3490488d/0.0.0/69546
58bab30a358235fa864b05cf819af0e179325c740e4bc853bcc7ec513e1...
Downloading data files:   0%|          | 0/3 [00:00<?, ?it/s]
Downloading data:   0%|          | 0.00/11.3M [00:00<?, ?B/s]
Downloading data:   0%|          | 0.00/1.35M [00:00<?, ?B/s]
Downloading data:   0%|          | 0.00/442k [00:00<?, ?B/s]
Extracting data files:   0%|          | 0/3 [00:00<?, ?it/s]
Generating train split: 0 examples [00:00, ? examples/s]
Generating test split: 0 examples [00:00, ? examples/s]
Generating validation split: 0 examples [00:00, ? examples/s]
Dataset csv downloaded and prepared to /root/.cache/huggingface/datasets/knkarthick___csv/knkarthick--dialogsum-cd36827d3490488d/0.0.0/6954658bab30a358235fa864b05cf
819af0e179325c740e4bc853bcc7ec513e1. Subsequent calls will reuse this data.
  0%|          | 0/3 [00:00<?, ?it/s]
```

Print a couple of dialogues with their baseline summaries.

In [6]:
```
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 [7]:
```
model_name='google/flan-t5-base'

model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
```

```
Downloading config.json:   0%|          | 0.00/1.40k [00:00<?, ?B/s]
Downloading pytorch_model.bin:   0%|          | 0.00/990M [00:00<?, ?B/s]
Downloading 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 [8]:
```
tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=True)
```

```
Downloading tokenizer_config.json:   0%|          | 0.00/2.54k [00:00<?, ?B/s]
Downloading spiece.model:   0%|          | 0.00/792k [00:00<?, ?B/s]
Downloading tokenizer.json:   0%|          | 0.00/2.42M [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/2.20k [00:00<?, ?B/s]
```

Test the tokenizer encoding and decoding a simple sentence:

In [9]:
```
sentence = "What time is it, Tom?"

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

sentence_decoded = tokenizer.decode(
        sentence_encoded["input_ids"][0],
        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?
```

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 [10]:
```
# MODEL WITHOUT PROMPT ENGINEERING
for i, index in enumerate(example_indices):
    dialogue = dataset['test'][index]['dialogue'] # Extracting dialogue from test dataset
    summary = dataset['test'][index]['summary'] # Extract corresponding summary from test dataset (Baseline Human Summary)

    # TOKENIZE INPUT DIALOGUE TO THE MODEL
    inputs = tokenizer(dialogue, return_tensors='pt')

    # GENERATING SUMMARY USING TOKENIZED INPUTS
    output = tokenizer.decode(
        model.generate(
            inputs["input_ids"],
            max_new_tokens=50, # CONSTRAINT ON NEW NO. TOKENS
        )[0],
        skip_special_tokens=True
    )

    # DISPLAY RESULTS
    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 [11]:
```
# ZERO-SHOT INFERENCE EXAMPLE - CONVERT DIALOGUE INTO AN INSTRUCTION PROMPT
# i.e. DESCRIBE THE EXPECTED OUTPUT OF THE MODEL (SUMMARIZE THE CONVERSATION)

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

    # PROMPT PROVIDES THE INSTRUCTION TO THE MODEL
    prompt = f"""
Summarize the following conversation.

{dialogue}

Summary:
    """
    # TOKENIZE CONSTRUCTED PROMPT INSTEAD OF THE DIALOGUE
    inputs = tokenizer(prompt, return_tensors='pt')

    # GENERATE SUMMARY USING THE MODEL // DECODE GENERATED TOKENS INTO STRINGS
    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:

Summarize the following conversation.

#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 train is about to leave.

---------------------------------------------------------------------------------------------------
Example  2
---------------------------------------------------------------------------------------------------
INPUT PROMPT:

Summarize the following conversation.

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

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 [13]:
```
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')
```

```
---------------------------------------------------------------------------------------------------
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 [14]:
```python
def make_prompt(example_indices_full, example_index_to_summarize):
    prompt = ''
    # Loop through the indices in example_indices_full. These examples provide context for the model.
    for index in example_indices_full:
        # Extract the dialogue and its corresponding human summary from the dataset.
        dialogue = dataset['test'][index]['dialogue']
        summary = dataset['test'][index]['summary']

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

{dialogue}

What was going on?
{summary}


"""
    # Extract the dialogue for the specific example to be summarized.
    dialogue = dataset['test'][example_index_to_summarize]['dialogue']

    # Append the dialogue of the example to be summarized to the prompt,
    # followed by the question "What was going on?" without providing a summary.
    # This sets up the model to generate a summary for this particular dialogue.
    prompt += f"""
Dialogue:

{dialogue}

What was going on?
"""

    return prompt
```

Construct the prompt to perform one shot inference:

In [15]:
```python
example_indices_full = [40] # ONE EXAMPLE -> ONE-SHOT INFERENCE
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 [16]:
```
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 [18]:
```
example_indices_full = [40, 80, 120] # THREE EXAMPLES -> FEW-SHOT INFERENCE
example_index_to_summarize = 200

few_shot_prompt = make_prompt(example_indices_full, example_index_to_summarize)

print(few_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#: May, do you mind helping me prepare for the picnic?
#Person2#: Sure. Have you checked the weather report?
#Person1#: Yes. It says it will be sunny all day. No sign of rain at all. This is your father's favorite sausage. Sandwiches for you and Daniel.
#Person2#: No, thanks Mom. I'd like some toast and chicken wings.
#Person1#: Okay. Please take some fruit salad and crackers for me.
#Person2#: Done. Oh, don't forget to take napkins disposable plates, cups and picnic blanket.
#Person1#: All set. May, can you help me take all these things to the living room?
#Person2#: Yes, madam.
#Person1#: Ask Daniel to give you a hand?
#Person2#: No, mom, I can manage it by myself. His help just causes more trouble.

What was going on?
Mom asks May to help to prepare for the picnic and May agrees.



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# wants to change the broken pendant in #Person2#'s shop.



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 [19]:
```python
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 i
n 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, if 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 [26]:
```
# generation_config = GenerationConfig(max_new_tokens=50)
# generation_config = GenerationConfig(max_new_tokens=10) # Limit of 10 new tokens, for shorter outputs.
# generation_config = GenerationConfig(max_new_tokens=50, do_sample=True, temperature=0.1) # Generates text with low randomness.
# 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) # High randomness in generation.

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:
He considers having Windows PC & CD system with painting feature of computers as well as graphics component in his current system - laptop computer. Also, person ha
s computer system working great.
---------------------------------------------------------------------------------------------------
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 [ ]:
```
```