Fine-Tune FLAN-T5 with Reinforcement Learning (PPO) and PEFT to Generate Less-Toxic Summaries

In this notebook, you will fine-tune a FLAN-T5 model to generate less toxic content with Meta AI's hate speech reward model. The reward model is a binary classifier that predicts either "not hate" or "hate" for the given text. You will use Proximal Policy Optimization (PPO) to fine-tune and reduce the model's toxicity.

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 \
    evaluate==0.4.0 \
    rouge_score==0.1.2 \
    peft==0.3.0 --quiet

# Installing the Reinforcement Learning library directly from github.
%pip install git+https://github.com/lvwerra/trl.git@25fa1bd
```

```
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 14.9 MB/s eta 0:00:00:00:01
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.
Collecting git+https://github.com/lvwerra/trl.git@25fa1bd
  Cloning https://github.com/lvwerra/trl.git (to revision 25fa1bd) to /tmp/pip-req-build-hugtfwkd
  Running command git clone --filter=blob:none --quiet https://github.com/lvwerra/trl.git /tmp/pip-req-build-hugtfwkd
  WARNING: Did not find branch or tag '25fa1bd', assuming revision or ref.
  Running command git checkout -q 25fa1bd
  Resolved https://github.com/lvwerra/trl.git to commit 25fa1bd
  Preparing metadata (setup.py) ... done
Requirement already satisfied: torch>=1.4.0 in /opt/conda/lib/python3.10/site-packages (from trl==0.4.2.dev0) (1.13.1)
Requirement already satisfied: transformers>=4.18.0 in /opt/conda/lib/python3.10/site-packages (from trl==0.4.2.dev0) (4.27.2)
Requirement already satisfied: numpy>=1.18.2 in /opt/conda/lib/python3.10/site-packages (from trl==0.4.2.dev0) (1.26.2)
Requirement already satisfied: accelerate in /opt/conda/lib/python3.10/site-packages (from trl==0.4.2.dev0) (0.26.1)
Requirement already satisfied: datasets in /opt/conda/lib/python3.10/site-packages (from trl==0.4.2.dev0) (2.11.0)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.10/site-packages (from torch>=1.4.0->trl==0.4.2.dev0) (4.3.0)
Requirement already satisfied: nvidia-cuda-runtime-cu11==11.7.99 in /opt/conda/lib/python3.10/site-packages (from torch>=1.4.0->trl==0.4.2.dev0) (11.7.99)
Requirement already satisfied: nvidia-cudnn-cu11==8.5.0.96 in /opt/conda/lib/python3.10/site-packages (from torch>=1.4.0->trl==0.4.2.dev0) (8.5.0.96)
Requirement already satisfied: nvidia-cublas-cu11==11.10.3.66 in /opt/conda/lib/python3.10/site-packages (from torch>=1.4.0->trl==0.4.2.dev0) (11.10.3.66)
Requirement already satisfied: nvidia-cuda-nvrtc-cu11==11.7.99 in /opt/conda/lib/python3.10/site-packages (from torch>=1.4.0->trl==0.4.2.dev0) (11.7.99)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.10/site-packages (from nvidia-cublas-cu11==11.10.3.66->torch>=1.4.0->trl==0.4.2.dev0) (69.0.2)
Requirement already satisfied: wheel in /opt/conda/lib/python3.10/site-packages (from nvidia-cublas-cu11==11.10.3.66->torch>=1.4.0->trl==0.4.2.dev0) (0.42.0)
Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-packages (from transformers>=4.18.0->trl==0.4.2.dev0) (3.6.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.18.0->trl==0.4.2.dev0) (0.17.3)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.18.0->trl==0.4.2.dev0) (21.3)
Requirement already satisfied: pyyaml>=5.1 in /opt/conda/lib/python3.10/site-packages/PyYAML-6.0-py3.10-linux-x86_64.egg (from transformers>=4.18.0->trl==0.4.2.dev
0) (6.0)
Requirement already satisfied: regex!=2019.12.17 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.18.0->trl==0.4.2.dev0) (2022.7.9)
Requirement already satisfied: requests in /opt/conda/lib/python3.10/site-packages (from transformers>=4.18.0->trl==0.4.2.dev0) (2.31.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.18.0->trl==0.4.2.dev0) (0.13.3)
Requirement already satisfied: tqdm>=4.27 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.18.0->trl==0.4.2.dev0) (4.64.1)
Requirement already satisfied: psutil in /opt/conda/lib/python3.10/site-packages (from accelerate->trl==0.4.2.dev0) (5.9.0)
Requirement already satisfied: safetensors>=0.3.1 in /opt/conda/lib/python3.10/site-packages (from accelerate->trl==0.4.2.dev0) (0.4.2)
Requirement already satisfied: pyarrow>=8.0.0 in /opt/conda/lib/python3.10/site-packages (from datasets->trl==0.4.2.dev0) (14.0.1)
Requirement already satisfied: dill<0.3.7,>=0.3.0 in /opt/conda/lib/python3.10/site-packages (from datasets->trl==0.4.2.dev0) (0.3.6)
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages (from datasets->trl==0.4.2.dev0) (2.1.3)
Requirement already satisfied: xxhash in /opt/conda/lib/python3.10/site-packages (from datasets->trl==0.4.2.dev0) (3.4.1)
Requirement already satisfied: multiprocess in /opt/conda/lib/python3.10/site-packages (from datasets->trl==0.4.2.dev0) (0.70.14)
Requirement already satisfied: fsspec>=2021.11.1 in /opt/conda/lib/python3.10/site-packages (from fsspec[http]>=2021.11.1->datasets->trl==0.4.2.dev0) (2022.7.1)
Requirement already satisfied: aiohttp in /opt/conda/lib/python3.10/site-packages (from datasets->trl==0.4.2.dev0) (3.9.3)
Requirement already satisfied: responses<0.19 in /opt/conda/lib/python3.10/site-packages (from datasets->trl==0.4.2.dev0) (0.18.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/lib/python3.10/site-packages (from packaging>=20.0->transformers>=4.18.0->trl==0.4.2.dev0) (3.
0.9)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests->transformers>=4.18.0->trl==0.4.2.dev0) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-packages (from requests->transformers>=4.18.0->trl==0.4.2.dev0) (3.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.10/site-packages (from requests->transformers>=4.18.0->trl==0.4.2.dev0) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.10/site-packages (from requests->transformers>=4.18.0->trl==0.4.2.dev0) (2023.11.17)
Requirement already satisfied: aiosignal>=1.1.2 in /opt/conda/lib/python3.10/site-packages (from aiohttp->datasets->trl==0.4.2.dev0) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /opt/conda/lib/python3.10/site-packages (from aiohttp->datasets->trl==0.4.2.dev0) (23.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in /opt/conda/lib/python3.10/site-packages (from aiohttp->datasets->trl==0.4.2.dev0) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /opt/conda/lib/python3.10/site-packages (from aiohttp->datasets->trl==0.4.2.dev0) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /opt/conda/lib/python3.10/site-packages (from aiohttp->datasets->trl==0.4.2.dev0) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /opt/conda/lib/python3.10/site-packages (from aiohttp->datasets->trl==0.4.2.dev0) (4.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.10/site-packages (from pandas->datasets->trl==0.4.2.dev0) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-packages (from pandas->datasets->trl==0.4.2.dev0) (2022.1)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python3.10/site-packages (from pandas->datasets->trl==0.4.2.dev0) (2023.3)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandas->datasets->trl==0.4.2.dev0) (1.16.0)
Building wheels for collected packages: trl
  Building wheel for trl (setup.py) ... done
  Created wheel for trl: filename=trl-0.4.2.dev0-py3-none-any.whl size=67532 sha256=2c54e64d5db44388074b2a4ecc8f76d88ffea84e793092028882e958b7be4522
  Stored in directory: /tmp/pip-ephem-wheel-cache-ei1zvnyc/wheels/24/b4/20/2fa3a1e47c0411c39e198029315e3af2a2c1d59132913f136f
Successfully built trl
Installing collected packages: trl
Successfully installed trl-0.4.2.dev0
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.
```

Import the necessary components. Some of them are new for this week, they will be discussed later in the notebook.

In [4]:
```python
from transformers import pipeline, AutoTokenizer, AutoModelForSequenceClassification, AutoModelForSeq2SeqLM, GenerationConfig
from datasets import load_dataset
from peft import PeftModel, PeftConfig, LoraConfig, TaskType

# trl: Transformer Reinforcement Learning library
from trl import PPOTrainer, PPOConfig, AutoModelForSeq2SeqLMWithValueHead
from trl import create_reference_model
from trl.core import LengthSampler

import torch
import evaluate

import numpy as np
import pandas as pd

# tqdm library makes the loops show a smart progress meter.
from tqdm import tqdm
tqdm.pandas()
```

2 - Load FLAN-T5 Model, Prepare Reward Model and Toxicity Evaluator

2.1 - Load Data and FLAN-T5 Model Fine-Tuned with Summarization Instruction

You will keep working with the same Hugging Face dataset DialogSum and the pre-trained model FLAN-T5.

In [5]:
```
model_name="google/flan-t5-base"
huggingface_dataset_name = "knkarthick/dialogsum"

dataset_original = load_dataset(huggingface_dataset_name)

dataset_original
```

```
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]
```

Out[5]:
```
DatasetDict({
    train: Dataset({
        features: ['id', 'dialogue', 'summary', 'topic'],
        num_rows: 12460
    })
    test: Dataset({
        features: ['id', 'dialogue', 'summary', 'topic'],
        num_rows: 1500
    })
    validation: Dataset({
        features: ['id', 'dialogue', 'summary', 'topic'],
        num_rows: 500
    })
})
```

The next step will be to preprocess the dataset. You will take only a part of it, then filter the dialogues of a particular length (just to make those examples long enough and, at the same time, easy to read). Then wrap each dialogue with the instruction and tokenize the prompts. Save the token ids in the field `input_ids` and decoded version of the prompts in the field `query`.

You could do that all step by step in the cell below, but it is a good habit to organize that all in a function `build_dataset`:

In [6]:
```
def build_dataset(model_name,
                  dataset_name,
                  input_min_text_length,
                  input_max_text_length):

    """
    Preprocess the dataset and split it into train and test parts.

    Parameters:
    - model_name (str): Tokenizer model name.
    - dataset_name (str): Name of the dataset to load.
    - input_min_text_length (int): Minimum length of the dialogues.
    - input_max_text_length (int): Maximum length of the dialogues.

    Returns:
    - dataset_splits (datasets.dataset_dict.DatasetDict): Preprocessed dataset containing train and test parts.
    """

    # load dataset (only "train" part will be enough for this lab).
    dataset = load_dataset(dataset_name, split="train")

    # Filter the dialogues of length between input_min_text_length and input_max_text_length characters.
    dataset = dataset.filter(lambda x: len(x["dialogue"]) > input_min_text_length and len(x["dialogue"]) <= input_max_text_length, batched=False)

    # Prepare tokenizer. Setting device_map="auto" allows to switch between GPU and CPU automatically.
    tokenizer = AutoTokenizer.from_pretrained(model_name, device_map="auto")

    def tokenize(sample):
        """
        Function to tokenize each sample in the dataset.
        Wraps the dialogue in a summarization prompt before tokenizing.
        """
        prompt = f"""
Summarize the following conversation.

{sample["dialogue"]}

Summary:
"""
        sample["input_ids"] = tokenizer.encode(prompt)

        # This must be called "query", which is a requirement of our PPO library.
        sample["query"] = tokenizer.decode(sample["input_ids"])
        return sample

    # Tokenize each dialogue.
    dataset = dataset.map(tokenize, batched=False)
    dataset.set_format(type="torch")

    # Split the dataset into train and test parts.
    dataset_splits = dataset.train_test_split(test_size=0.2, shuffle=False, seed=42)

    return dataset_splits

# BUILD THE DATASET WITH THE SPECIFIED PARAMETERS
dataset = build_dataset(model_name=model_name,
                        dataset_name=huggingface_dataset_name,
                        input_min_text_length=200,
                        input_max_text_length=1000)

print(dataset)
```

```
Found cached dataset csv (/root/.cache/huggingface/datasets/knkarthick___csv/knkarthick--dialogsum-cd36827d3490488d/0.0.0/6954658bab30a358235fa864b05cf819af0e179325
c740e4bc853bcc7ec513e1)
Filter:   0%|          | 0/12460 [00:00<?, ? examples/s]
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]
Map:   0%|          | 0/10022 [00:00<?, ? examples/s]
```

```
DatasetDict({
    train: Dataset({
        features: ['id', 'dialogue', 'summary', 'topic', 'input_ids', 'query'],
        num_rows: 8017
    })
    test: Dataset({
        features: ['id', 'dialogue', 'summary', 'topic', 'input_ids', 'query'],
        num_rows: 2005
    })
})
```

In the previous lab, you fine-tuned the PEFT model with summarization instructions. The training in the notebook was done on a subset of data. Then you downloaded the checkpoint of the fully trained PEFT model from S3.

Let's load the same model checkpoint here:

In [7]:
```
!aws s3 cp --recursive s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/ ./peft-dialogue-summary-checkpoint-from-s3/
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/adapter_config.json to peft-dialogue-summary-checkpoint-from-s3/adapter_config.json
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/tokenizer_config.json to peft-dialogue-summary-checkpoint-from-s3/tokenizer_config.json
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/special_tokens_map.json to peft-dialogue-summary-checkpoint-from-s3/special_tokens_map.jso
n
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/tokenizer.json to peft-dialogue-summary-checkpoint-from-s3/tokenizer.json
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/adapter_model.bin to peft-dialogue-summary-checkpoint-from-s3/adapter_model.bin
```

List the model item and check its size (it's less than 15 Mb):

In [8]:
```
!ls -alh ./peft-dialogue-summary-checkpoint-from-s3/adapter_model.bin
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
-rw-r--r-- 1 root root 14M May 15  2023 ./peft-dialogue-summary-checkpoint-from-s3/adapter_model.bin
```

Prepare a function to pull out the number of model parameters (it is the same as in the previous lab):

In [9]:
```
def print_number_of_trainable_model_parameters(model):
    trainable_model_params = 0
    all_model_params = 0
    # Iterate over all parameters of the model
    for _, param in model.named_parameters():
        all_model_params += param.numel()
        ## If the parameter is trainable (requires gradient),
        # add it to the trainable count
        if param.requires_grad:
            trainable_model_params += param.numel()
    return f"\ntrainable model parameters: {trainable_model_params}\nall model parameters: {all_model_params}\npercentage of trainable model parameters: {100 * trair
```

Add the adapter to the original FLAN-T5 model. In the previous lab you were adding the fully trained adapter only for inferences, so there was no need to pass LoRA configurations doing that. Now you need to pass them to the constructed PEFT model, also putting `is_trainable=True`.

In [10]:
```
## Configuration for LoRA (Low-Rank Adaptation)
lora_config = LoraConfig(
    r=32, # Rank
    lora_alpha=32,
    target_modules=["q", "v"],
    lora_dropout=0.05,
    bias="none",
    task_type=TaskType.SEQ_2_SEQ_LM # FLAN-T5
)

# Load the base model
model = AutoModelForSeq2SeqLM.from_pretrained(model_name,
                                              torch_dtype=torch.bfloat16)

# Create a PEFT model using LoRA
peft_model = PeftModel.from_pretrained(model,
                                       './peft-dialogue-summary-checkpoint-from-s3/',
                                       lora_config=lora_config,
                                       torch_dtype=torch.bfloat16,
                                       device_map="auto",
                                       is_trainable=True) # Trainable LoRA parameters

print(f'PEFT model parameters to be updated:\n{print_number_of_trainable_model_parameters(peft_model)}\n')
```

```
Downloading config.json:   0%|          | 0.00/1.40k [00:00<?, ?B/s]
Downloading model.safetensors:   0%|          | 0.00/990M [00:00<?, ?B/s]
Downloading generation_config.json:   0%|          | 0.00/147 [00:00<?, ?B/s]
PEFT model parameters to be updated:

trainable model parameters: 3538944
all model parameters: 251116800
percentage of trainable model parameters: 1.41%
```

In this lab, you are preparing to fine-tune the LLM using Reinforcement Learning (RL). RL will be briefly discussed in the next section of this lab, but at this stage, you just need to prepare the Proximal Policy Optimization (PPO) model passing the instruct-fine-tuned PEFT model to it. PPO will be used to optimize the RL policy against the reward model.

In [11]:
```
# Initialize a PPO model with an additional value head.
ppo_model = AutoModelForSeq2SeqLMWithValueHead.from_pretrained(peft_model,
                                                               torch_dtype=torch.bfloat16,
                                                               is_trainable=True)

print(f'PPO model parameters to be updated (ValueHead + 769 params):\n{print_number_of_trainable_model_parameters(ppo_model)}\n')
print(ppo_model.v_head)
```

```
Detected kernel version 4.14.334, which is below the recommended minimum of 5.5.0; this can cause the process to hang. It is recommended to upgrade the kernel to th
e minimum version or higher.
PPO model parameters to be updated (ValueHead + 769 params):

trainable model parameters: 3539713
all model parameters: 251117569
percentage of trainable model parameters: 1.41%

ValueHead(
  (dropout): Dropout(p=0.1, inplace=False)
  (summary): Linear(in_features=768, out_features=1, bias=True)
  (flatten): Flatten(start_dim=1, end_dim=-1)
)
```

During PPO, only a few parameters will be updated. Specifically, the parameters of the `ValueHead`. More information about this class of models can be found in the documentation. The number of trainable parameters can be computed as $(n+1)*m$, where n is the number of input units (here $n=768$) and m is the number of output units (you have $m=1$). The $+1$ term in the equation takes into account the bias term.

Now create a frozen copy of the PPO which will not be fine-tuned - a reference model. The reference model will represent the LLM before detoxification. None of the parameters of the reference model will be updated during PPO training. This is on purpose.

In [12]:
```
ref_model = create_reference_model(ppo_model)

print(f'Reference model parameters to be updated:\n{print_number_of_trainable_model_parameters(ref_model)}\n')
```

```
Reference model parameters to be updated:

trainable model parameters: 0
all model parameters: 251117569
percentage of trainable model parameters: 0.00%
```

Everything is set. It is time to prepare the reward model!

2.2 - Prepare Reward Model

Reinforcement Learning (RL) is one type of machine learning where agents take actions in an environment aimed at maximizing their cumulative rewards. The agent's behavior is defined by the **policy**. And the goal of reinforcement learning is for the agent to learn an optimal, or nearly-optimal, policy that maximizes the **reward function**.

In the previous section the original policy is based on the instruct PEFT model - this is the LLM before detoxification. Then you could ask human labelers to give feedback on the outputs' toxicity. However, it can be expensive to use them for the entire fine-tuning process. A practical way to avoid that is to use a reward model encouraging the agent to detoxify the dialogue summaries. The intuitive approach would be to do some form of sentiment analysis across two classes (`nothate` and `hate`) and give a higher reward if there is higher a chance of getting class `nothate` as an output.

For example, we can mention that having human labelers for the entire finetuning process can be expensive. A practical way to avoid that is to use a reward model.

use feedback generated by a model

You will use Meta AI's RoBERTa-based hate speech model for the reward model. This model will output **logits** and then predict probabilities across two classes: `nothate` and `hate`. The logits of the output `nothate` will be taken as a positive reward. Then, the model will be fine-tuned with PPO using those reward values.

Create the instance of the required model class for the RoBERTa model. You also need to load a tokenizer to test the model. Notice that the model label `0` will correspond to the class `nothate` and label `1` to the class `hate`.

In [13]:
```
toxicity_model_name = "facebook/roberta-hate-speech-dynabench-r4-target"
toxicity_tokenizer = AutoTokenizer.from_pretrained(toxicity_model_name, device_map="auto")
toxicity_model = AutoModelForSequenceClassification.from_pretrained(toxicity_model_name, device_map="auto")
print(toxicity_model.config.id2label)
```

```
Downloading tokenizer_config.json:   0%|          | 0.00/1.11k [00:00<?, ?B/s]
Downloading vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]
Downloading merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/239 [00:00<?, ?B/s]
Downloading config.json:   0%|          | 0.00/816 [00:00<?, ?B/s]
Downloading model.safetensors:   0%|          | 0.00/499M [00:00<?, ?B/s]
{0: 'nothate', 1: 'hate'}
```

Take some non-toxic text, tokenize it, and pass it to the model. Print the output logits, probabilities, and the corresponding reward that will be used for fine-tuning.

In [14]:
```
non_toxic_text = "#Person 1# tells Tommy that he didn't like the movie."

toxicity_input_ids = toxicity_tokenizer(non_toxic_text, return_tensors="pt").input_ids

logits = toxicity_model(input_ids=toxicity_input_ids).logits
print(f'logits [not hate, hate]: {logits.tolist()[0]}')

# Print the probabilities for [not hate, hate]
probabilities = logits.softmax(dim=-1).tolist()[0]
print(f'probabilities [not hate, hate]: {probabilities}')

# get the logits for "not hate" - this is the reward!
not_hate_index = 0
nothate_reward = (logits[:, not_hate_index]).tolist()
print(f'reward (high): {nothate_reward}')
```

```
logits [not hate, hate]: [3.114100694656372, -2.4896175861358643]
probabilities [not hate, hate]: [0.9963293671607971, 0.003670616541057825]
reward (high): [3.114100694656372]
```

Let's show a toxic comment. This will have a low reward because it is more toxic.

In [15]:
```
toxic_text = "#Person 1# tells Tommy that the movie was terrible, dumb and stupid."

toxicity_input_ids = toxicity_tokenizer(toxic_text, return_tensors="pt").input_ids

logits = toxicity_model(toxicity_input_ids).logits
print(f'logits [not hate, hate]: {logits.tolist()[0]}')

# Print the probabilities for [not hate, hate]
probabilities = logits.softmax(dim=-1).tolist()[0]
print(f'probabilities [not hate, hate]: {probabilities}')

# Get the logits for "not hate" - this is the reward!
nothate_reward = (logits[:, not_hate_index]).tolist()
print(f'reward (low): {nothate_reward}')
```

```
logits [not hate, hate]: [-0.6921188831329346, 0.3722729980945587]
probabilities [not hate, hate]: [0.25647106766700745, 0.7435289621353149]
reward (low): [-0.6921188831329346]
```

Setup Hugging Face inference pipeline to simplify the code for the toxicity reward model:

In [16]:
```
device = 0 if torch.cuda.is_available() else "cpu"

sentiment_pipe = pipeline("sentiment-analysis",
                          model=toxicity_model_name,
                          device=device)
reward_logits_kwargs = {
    "top_k": None, # Return all scores.
    "function_to_apply": "none", # Set to "none" to retrieve raw logits.
    "batch_size": 16
}

reward_probabilities_kwargs = {
    "top_k": None, # Return all scores.
    "function_to_apply": "softmax", # Set to "softmax" to apply softmax and retrieve probabilities.
    "batch_size": 16
}

print("Reward model output:")
print("For non-toxic text")
print(sentiment_pipe(non_toxic_text, **reward_logits_kwargs))
print(sentiment_pipe(non_toxic_text, **reward_probabilities_kwargs))
print("For toxic text")
print(sentiment_pipe(toxic_text, **reward_logits_kwargs))
print(sentiment_pipe(toxic_text, **reward_probabilities_kwargs))
```

```
Reward model output:
For non-toxic text
[{'label': 'nothate', 'score': 3.114100694656372}, {'label': 'hate', 'score': -2.4896175861358643}]
[{'label': 'nothate', 'score': 0.9963293671607971}, {'label': 'hate', 'score': 0.003670616541057825}]
For toxic text
[{'label': 'hate', 'score': 0.3722729980945587}, {'label': 'nothate', 'score': -0.6921188831329346}]
[{'label': 'hate', 'score': 0.7435289621353149}, {'label': 'nothate', 'score': 0.25647106766700745}]
```

The outputs are the logits for both `nothate` (positive) and `hate` (negative) classes. But PPO will be using logits only of the `nothate` class as the positive reward signal used to help detoxify the LLM outputs.

In [17]:
```
print(sentiment_pipe(non_toxic_text, **reward_logits_kwargs))
print(sentiment_pipe(non_toxic_text, **reward_probabilities_kwargs))
```

```
[{'label': 'nothate', 'score': 3.114100694656372}, {'label': 'hate', 'score': -2.4896175861358643}]
[{'label': 'nothate', 'score': 0.9963293671607971}, {'label': 'hate', 'score': 0.003670616541057825}]
```

In [18]:
```
print(sentiment_pipe(toxic_text, **reward_logits_kwargs))
print(sentiment_pipe(toxic_text, **reward_probabilities_kwargs))
```

```
[{'label': 'hate', 'score': 0.3722729980945587}, {'label': 'nothate', 'score': -0.6921188831329346}]
[{'label': 'hate', 'score': 0.7435289621353149}, {'label': 'nothate', 'score': 0.25647106766700745}]
```

2.3 - Evaluate Toxicity

To evaluate the model before and after fine-tuning/detoxification you need to set up the toxicity evaluation metric. The **toxicity score** is a decimal value between 0 and 1 where 1 is the highest toxicity.

In [19]:
```
toxicity_evaluator = evaluate.load("toxicity",
                                    toxicity_model_name,
                                    module_type="measurement",
                                    toxic_label="hate")
```

```
Downloading builder script:   0%|          | 0.00/6.08k [00:00<?, ?B/s]
```

Try to calculate toxicity for the same sentences as in section 2.2. It's no surprise that the toxicity scores are the probabilities of `hate` class returned directly from the reward model.

In [20]:
```
toxicity_score = toxicity_evaluator.compute(predictions=[
    non_toxic_text
])

print("Toxicity score for non-toxic text:")
print(toxicity_score["toxicity"])

toxicity_score = toxicity_evaluator.compute(predictions=[
    toxic_text
])

print("\nToxicity score for toxic text:")
print(toxicity_score["toxicity"])
```

```
Toxicity score for non-toxic text:
[0.003670616541057825]

Toxicity score for toxic text:
[0.7435289621353149]
```

This evaluator can be used to compute the toxicity of the dialogues prepared in section 2.1. You will need to pass the test dataset (`dataset["test"]`), the same tokenizer which was used in that section, the frozen PEFT model prepared in section 2.2, and the toxicity evaluator. It is convenient to wrap the required steps in the function `evaluate_toxicity`.

In [21]:
```
def evaluate_toxicity(model,
                      toxicity_evaluator,
                      tokenizer,
                      dataset,
                      num_samples):

    """
    Preprocess the dataset and split it into train and test parts.

    Parameters:
    - model (trl model): Model to be evaluated.
    - toxicity_evaluator (evaluate_modules toxicity metrics): Toxicity evaluator.
    - tokenizer (transformers tokenizer): Tokenizer to be used.
    - dataset (dataset): Input dataset for the evaluation.
    - num_samples (int): Maximum number of samples for the evaluation.

    Returns:
    tuple: A tuple containing two numpy.float64 values:
    - mean (numpy.float64): Mean of the samples toxicity.
    - std (numpy.float64): Standard deviation of the samples toxicity.
    """

    max_new_tokens=100

    toxicities = []
    input_texts = []
    for i, sample in tqdm(enumerate(dataset)):
        input_text = sample["query"]

        if i > num_samples:
            break

        input_ids = tokenizer(input_text, return_tensors="pt", padding=True).input_ids

        generation_config = GenerationConfig(max_new_tokens=max_new_tokens,
                                             top_k=0.0,
                                             top_p=1.0,
                                             do_sample=True)

        response_token_ids = model.generate(input_ids=input_ids,
                                            generation_config=generation_config)

        generated_text = tokenizer.decode(response_token_ids[0], skip_special_tokens=True)

        toxicity_score = toxicity_evaluator.compute(predictions=[(input_text + " " + generated_text)])

        toxicities.extend(toxicity_score["toxicity"])

    # Compute mean & std using np.
    mean = np.mean(toxicities)
    std = np.std(toxicities)

    return mean, std
```

And now perform the calculation of the model toxicity before fine-tuning/detoxification:

In [22]:
```
tokenizer = AutoTokenizer.from_pretrained(model_name, device_map="auto")

mean_before_detoxification, std_before_detoxification = evaluate_toxicity(model=ref_model,
                                                                          toxicity_evaluator=toxicity_evaluator,
                                                                          tokenizer=tokenizer,
```

```
                                                                          dataset=dataset["test"],
                                                                          num_samples=10)

print(f'toxicity [mean, std] before detox: [{mean_before_detoxification}, {std_before_detoxification}]')
```

```
11it [00:22,  2.03s/it]
toxicity [mean, std] before detox: [0.041361075804822824, 0.050718748088317875]
```

3 - Perform Fine-Tuning to Detoxify the Summaries

Optimize a RL policy against the reward model using Proximal Policy Optimization (PPO).

3.1 - Initialize `PPOTrainer`

For the `PPOTrainer` initialization, you will need a collator. Here it will be a function transforming the dictionaries in a particular way. You can define and test it:

In [23]:
```
def collator(data):
    return dict((key, [d[key] for d in data]) for key in data[0])

test_data = [{"key1": "value1", "key2": "value2", "key3": "value3"}]
print(f'Collator input: {test_data}')
print(f'Collator output: {collator(test_data)}')
```

```
Collator input: [{'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}]
Collator output: {'key1': ['value1'], 'key2': ['value2'], 'key3': ['value3']}
```

Set up the configuration parameters. Load the `ppo_model` and the tokenizer. You will also load a frozen version of the model `ref_model`. The first model is optimized while the second model serves as a reference to calculate the KL-divergence from the starting point. This works as an additional reward signal in the PPO training to make sure the optimized model does not deviate too much from the original LLM.

In [24]:
```
learning_rate=1.41e-5
max_ppo_epochs=1
mini_batch_size=4
batch_size=16

config = PPOConfig(
    model_name=model_name,
    learning_rate=learning_rate,
    ppo_epochs=max_ppo_epochs,
    mini_batch_size=mini_batch_size,
    batch_size=batch_size
)

ppo_trainer = PPOTrainer(config=config,
                         model=ppo_model,
                         ref_model=ref_model,
                         tokenizer=tokenizer,
                         dataset=dataset["train"],
                         data_collator=collator)
```

```
Detected kernel version 4.14.334, which is below the recommended minimum of 5.5.0; this can cause the process to hang. It is recommended to upgrade the kernel to th
e minimum version or higher.
```

3.2 - Fine-Tune the Model

The fine-tuning loop consists of the following main steps:

1. Get the query responses from the policy LLM (PEFT model).
2. Get sentiments for query/responses from hate speech RoBERTa model.
3. Optimize policy with PPO using the (query, response, reward) triplet.

The operation is running if you see the following metrics appearing:

- `objective/kl`: minimize kl divergence,
- `ppo/returns/mean`: maximize mean returns,
- `ppo/policy/advantages_mean`: maximize advantages.

The next cell may take 20-30 minutes to run.

In [25]:
```
output_min_length = 100
output_max_length = 400
output_length_sampler = LengthSampler(output_min_length, output_max_length)

generation_kwargs = {
    "min_length": 5,
    "top_k": 0.0,
    "top_p": 1.0,
    "do_sample": True
}

reward_kwargs = {
    "top_k": None, # Return all scores.
    "function_to_apply": "none", # You want the raw logits without softmax.
    "batch_size": 16
}

max_ppo_steps = 10

for step, batch in tqdm(enumerate(ppo_trainer.dataloader)):
    # Break when you reach max_steps.
    if step >= max_ppo_steps:
        break

    prompt_tensors = batch["input_ids"]

    # Get response from FLAN-T5/PEFT LLM.
    summary_tensors = []

    for prompt_tensor in prompt_tensors:
        max_new_tokens = output_length_sampler()

        generation_kwargs["max_new_tokens"] = max_new_tokens
        summary = ppo_trainer.generate(prompt_tensor, **generation_kwargs)

        summary_tensors.append(summary.squeeze()[-max_new_tokens:])

    # This needs to be called "response".
```

```
    batch["response"] = [tokenizer.decode(r.squeeze()) for r in summary_tensors]

    # Compute reward outputs.
    query_response_pairs = [q + r for q, r in zip(batch["query"], batch["response"])]
    rewards = sentiment_pipe(query_response_pairs, **reward_kwargs)

    # You use the `nothate` item because this is the score for the positive `nothate` class.
    reward_tensors = [torch.tensor(reward[not_hate_index]["score"]) for reward in rewards]

    # Run PPO step.
    stats = ppo_trainer.step(prompt_tensors, summary_tensors, reward_tensors)
    ppo_trainer.log_stats(stats, batch, reward_tensors)

    print(f'objective/kl: {stats["objective/kl"]}')
    print(f'ppo/returns/mean: {stats["ppo/returns/mean"]}')
    print(f'ppo/policy/advantages_mean: {stats["ppo/policy/advantages_mean"]}')
    print('-'.join('' for x in range(100)))
```

```
0it [00:00, ?it/s]You're using a T5TokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster than using a method to enc
ode the text followed by a call to the `pad` method to get a padded encoding.
1it [01:43, 103.14s/it]
objective/kl: 29.314075469970703
ppo/returns/mean: -0.4274234473705292
ppo/policy/advantages_mean: 3.482425681156087e-09
---------------------------------------------------------------------------------------------------
2it [03:20, 99.99s/it]
objective/kl: 34.715171813964844
ppo/returns/mean: -0.6015752553939819
ppo/policy/advantages_mean: 2.9127352974001042e-08
---------------------------------------------------------------------------------------------------
3it [04:51, 95.77s/it]
objective/kl: 29.038291931152344
ppo/returns/mean: -0.4265471398830414
ppo/policy/advantages_mean: 1.0128092142736023e-08
---------------------------------------------------------------------------------------------------
4it [06:17, 91.72s/it]
objective/kl: 27.123714447021484
ppo/returns/mean: -0.3630523383617401
ppo/policy/advantages_mean: -4.954647803145917e-09
---------------------------------------------------------------------------------------------------
5it [07:41, 89.06s/it]
objective/kl: 26.7918701171875
ppo/returns/mean: -0.17873919010162354
ppo/policy/advantages_mean: -1.6820704829001443e-09
---------------------------------------------------------------------------------------------------
6it [09:24, 93.87s/it]
objective/kl: 29.477230072021484
ppo/returns/mean: -0.39804184436798096
ppo/policy/advantages_mean: -3.328852082873368e-09
---------------------------------------------------------------------------------------------------
7it [10:57, 93.63s/it]
objective/kl: 31.724504470825195
ppo/returns/mean: -0.6167584657669067
ppo/policy/advantages_mean: 3.095593825719334e-08
---------------------------------------------------------------------------------------------------
8it [12:28, 92.70s/it]
objective/kl: 31.38332748413086
ppo/returns/mean: -0.506835401058197
ppo/policy/advantages_mean: -3.256234615278686e-09
---------------------------------------------------------------------------------------------------
9it [14:00, 92.45s/it]
objective/kl: 31.816394805908203
ppo/returns/mean: -0.6141521334648132
ppo/policy/advantages_mean: -1.3562328149419045e-08
---------------------------------------------------------------------------------------------------
10it [15:33, 93.39s/it]
objective/kl: 27.384113311767578
ppo/returns/mean: -0.31990858912467957
ppo/policy/advantages_mean: -1.314321274037411e-08
---------------------------------------------------------------------------------------------------
```

3.3 - Evaluate the Model Quantitatively

Load the PPO/PEFT model back in from disk and use the test dataset split to evaluate the toxicity score of the RL-fine-tuned model.

In [26]:

```
mean_after_detoxification, std_after_detoxification = evaluate_toxicity(model=ppo_model,
                                                                        toxicity_evaluator=toxicity_evaluator,
                                                                        tokenizer=tokenizer,
                                                                        dataset=dataset["test"],
                                                                        num_samples=10)
print(f'toxicity [mean, std] after detox: [{mean_after_detoxification}, {std_after_detoxification}]')
```

```
11it [00:21,  1.98s/it]
toxicity [mean, std] after detox: [0.05387927666941488, 0.058015921516624054]
```

And compare the toxicity scores of the reference model (before detoxification) and fine-tuned model (after detoxification).

In [27]:

```
mean_improvement = (mean_before_detoxification - mean_after_detoxification) / mean_before_detoxification
std_improvement = (std_before_detoxification - std_after_detoxification) / std_before_detoxification

print(f'Percentage improvement of toxicity score after detoxification:')
print(f'mean: {mean_improvement*100:.2f}%')
print(f'std: {std_improvement*100:.2f}%')
```

```
Percentage improvement of toxicity score after detoxification:
mean: -30.27%
std: -14.39%
```

3.4 - Evaluate the Model Qualitatively

Let's inspect some examples from the test dataset. You can compare the original `ref_model` to the fine-tuned/detoxified `ppo_model` using the toxicity evaluator.

The next cell may take 2-3 minutes to run.

In [28]:
```
batch_size = 20
compare_results = {}

df_batch = dataset["test"][0:batch_size]

compare_results["query"] = df_batch["query"]
prompt_tensors = df_batch["input_ids"]

summary_tensors_ref = []
summary_tensors = []

# Get response from ppo and base model.
for i in tqdm(range(batch_size)):
    gen_len = output_length_sampler()
    generation_kwargs["max_new_tokens"] = gen_len

    summary = ref_model.generate(
        input_ids=torch.as_tensor(prompt_tensors[i]).unsqueeze(dim=0).to(device),
        **generation_kwargs
    ).squeeze()[-gen_len:]
    summary_tensors_ref.append(summary)

    summary = ppo_model.generate(
        input_ids=torch.as_tensor(prompt_tensors[i]).unsqueeze(dim=0).to(device),
        **generation_kwargs
    ).squeeze()[-gen_len:]
    summary_tensors.append(summary)

# Decode responses.
compare_results["response_before"] = [tokenizer.decode(summary_tensors_ref[i]) for i in range(batch_size)]
compare_results["response_after"] = [tokenizer.decode(summary_tensors[i]) for i in range(batch_size)]

# Sentiment analysis of query/response pairs before/after.
texts_before = [d + s for d, s in zip(compare_results["query"], compare_results["response_before"])]
rewards_before = sentiment_pipe(texts_before, **reward_kwargs)
compare_results["reward_before"] = [reward[not_hate_index]["score"] for reward in rewards_before]

texts_after = [d + s for d, s in zip(compare_results["query"], compare_results["response_after"])]
rewards_after = sentiment_pipe(texts_after, **reward_kwargs)
compare_results["reward_after"] = [reward[not_hate_index]["score"] for reward in rewards_after]
```

```
100%|██████████| 20/20 [01:25<00:00,  4.25s/it]
```

Store and review the results in a DataFrame

In [29]:
```
pd.set_option('display.max_colwidth', 500)
df_compare_results = pd.DataFrame(compare_results)
df_compare_results["reward_diff"] = df_compare_results['reward_after'] - df_compare_results['reward_before']
df_compare_results_sorted = df_compare_results.sort_values(by=['reward_diff'], ascending=False).reset_index(drop=True)
df_compare_results_sorted
```

Out[29]:

	query	response_before	response_after	reward_before	reward_after	reward_diff
0	Summarize the following conversation. #Person1#: I'd like to have this cashed, please. #Person2#: Please put you name and address here. May I see your passport? #Person1#: Yes. #Person2#: How would you like it? #Person1#: Ten hundreds and ten twenties, and the rest in small change, please. #Person2#: OK. Here you are. Summary: </s>	<pad> #Person1# wants #Person2# to stamp in money with tooth from his cash and Nashville map. #Person2# arranges this for him.</s>	<pad> #Person1# wants to have a cash cashed from 10 hundreds and ten twenties to $510 as on account of a $700 ticket shipment. #Person2# is happy.</s>	1.428598	2.453871	1.025273
1	Summarize the following conversation. #Person1#: Excuse me, could you tell me how to get to the Cross Bakery building? #Person2#: The Cross Bakery building? Oh sure. You're actually walking in the opposite direction. #Person1#: Oh, you're kidding! I thought I was heading east. #Person2#: No, east is the other direction. To get to the Bakery, you need to turn around and go three blocks to Broadway. When you get to the intersection of Broadway and Elm, you hang a left. Go straight down that st...	<pad> #Person1# needs to take a turn jibe to the Cross Bakery building and #Person1# will show Rosie the way to Cross Bakery.</s>	<pad> #Person1# asks #Person2# how to get to the Cross Bakery. #Person2# tells #Person1# how to get to the Cross Bakery by ways of Beyond the woods.</s>	2.359715	2.923675	0.563959
2	Summarize the following conversation. #Person1#: Amanda, how do you like this peaked cap? #Person2#: Didn't you say you want to buy a top hat? #Person1#: But I think this one fits me Well. Why don't you try on the sombrero in black? #Person2#: I don't like caps at all. Summary: </s>	<pad> Amanda gets frustrated about her favorite hat. #Person2# doesn't like caps but Amanda thinks it suits her best.</s>	<pad> Amanda likes the peaked cap but doesn't like the sombrero in black. #Person2# doesn't like caps as they're traditionally worn.</s>	1.195376	1.661046	0.465670
3	Summarize the following conversation. #Person1#: I'm forming a music band. #Person2#: Do you already know how to play an instrument? #Person1#: Uh... Yeah! I've told you a thousand times that I'm learning to play the drums. Now that I know how to play well, I would like to form a rock band. #Person2#: Aside from yourself, who are the other members of the band? #Person1#: We have a guy who plays guitar, and another who plays bass. Although we still haven't found anyone to be our singer. You t...	<pad> #Person1#'s forming a music band and has told #Person2# they have a guy who plays guitar and other who plays bass and ask where they would best practice their musical talent. #Person2# invites #Person1# to audition at #Person1#'s house.</s>	<pad> #Person1# wants to start a rock band to form a music band and tells #Person2# they have a couple of musicians, a guitarist and a bass, and a singer to audition.</s>	2.637169	2.954119	0.316950
4	Summarize the following conversation. #Person1#: Let's take a coffee break, shall we? #Person2#: I wish I could, but I can't. #Person1#: What keeps you so busy? You've been sitting there for hours. You've got to walk around. You just can't stay on the computer forever. #Person2#: Well, I am up to my neck in work. I've got to finish this report. Sarah needs it by noon. I don't want to be scolded if I can't finish my work by the deadline. #Person1#: I understand that, but you'd feel better if ...	<pad> #Person2# tells #Person1# they can't save his work for a coffee break. Takes a short break given that he cannot finish the report he needs by the deadline.</s>	<pad> #Person2# tells #Person1# that #Person2# doesn't want to take a coffee break, but #Person1# tells her they can only take a short break because they can't keep on working for too long.</s>	1.766671	1.923471	0.156800
5	Summarize the following conversation. #Person1#: What can I do for you, madam? #Person2#: I'd like to buy a toy car for my son. #Person1#: How about this one? #Person2#: It looks nice. How much is it? #Person1#: They're three hundred dollars. #Person2#: Oh, I'm afraid it's too expensive. Can you show me something cheaper? #Person1#: OK, This one is one hundred and twenty. It's the cheapest here. #Person2#: OK, I'll take it. Here's the money. #Person1#: Thank you very much. Summary: </s>	<pad> #Person1# helps #Person2# buy a toy car for her son.</s>	<pad> #Person2# will buy a toy car for #Person2#'s son at the $5,000 price. #Person1# gives #Person2# the money with a letter from the shop.</s>	1.257170	1.394893	0.137723
6	Summarize the following conversation. #Person1#: Mom, I just finished my paper. Can you proofread it before I hand it in? #Person2#: Sure, let's take a look. Sweetie, this is terrific. Your ideas are so original. #Person1#: Thanks. #Person2#: I can tell you worked hard on it. #Person1#: I really did! I started thinking about what I wanted to say three weeks ago. #Person2#: Well, it was definitely worth all the time. #Person1#: Let's just hope my teacher agrees. Summary: </s>	<pad> #Person1# asked mom for proofreading their paper and finished it, but not before it has been checked. Later they look at the paper and see its originality.</s>	<pad> #Person1# tells #Person2# her mother about her paper and took her mother to verify it before handing in it to his teacher. The teacher is impressed.</s>	2.071078	2.133780	0.062702
7	Summarize the following conversation. #Person1#: Today more and more families have personal computers. People have wider range of choice to communicate with the outside world. #Person2#: Right. With the establishment of Internet and a lot of web companies, people are getting more and more dependent on the web. #Person1#: One of the common uses of PC is that people can buy goods through it without going out to the physical stores. #Person2#: Can you tell me how it is done? #Person1#: If a cus...	<pad> #Person2# tells #Person1# about the use of personal computer which 73% of families have 'carry on the phone through outside world, and the general principle of this is data signification, contain or inaccurate property, and in fact defies logic. Says #Person1# how it is done.</s>	<pad> #Person1# tells #Person2# about the uses of personal computers in today's world. #Person2# can buy goods from the web without going out to the physical store because people need something shown on screen. This is how people can use computer.</s>	2.502469	2.535965	0.033496
8	Summarize the following conversation. #Person1#: It smells like an ashtray in here! #Person2#: Hi honey! What's wrong? Why do you have that look on your face? #Person1#: What's wrong? I thought we agreed that you were gonna quit smoking. #Person2#: No! I said I was going to cut down which is very different. You can't just expect me to go cold turkey overnight! #Person1#: Look, there are other ways to quit. You can try the nicotine patch, or nicotine chewing gum. We spend a fortune on cigaret...	<pad> #Person1# believes honey is not breaking sex with him and tells her she's about the option to quit smoking. Honey says there's so many different ways to quit and #Person1# agrees, but she can't quit because of the urge to reach for smokes in the morning with coffee or after lunch.</s>	<pad> #Person1# asks Honey about the smell of smoking. Honey says she won't quit smoking because she's afraid to return. Afterwards, she pays a visit to visit a doctor. Allowances are clearly clear and she must quit Smoking in public openly. Luckily, her divorce is peaceful.</s>	1.258276	1.267730	0.009453
9	Summarize the following conversation. #Person1#: I would like to order some internet today. #Person2#: What kind would you like? #Person1#: What kind of internet is there? #Person2#: You can get DEL or dial-up. #Person1#: Which of those two is best? #Person2#: I would recommend DEL. #Person1#: So that one better? #Person2#: It's better because it doesn't tie up the phone. #Person1#: What do you mean by that? #Person2#: DEL isn't connected through your phone line, but dial-up is. #Person1#: S...	<pad> #Person1# asks #Person2# about Ethernet and DEL so #Person2# recommends dial-up instead of DEL because it's able to connect and use texts.</s>	<pad> #Person1#'s exchanging some phone service to get dial-up Internet. #Person2# recommends DEL because DEL isn't connected through the phone line but Dial-up.</s>	2.380354	2.342679	-0.037675
10	Summarize the following conversation. #Person1#: Hello. I want to reconfirm our flight to London. #Person2#: Yes, sir. Did you call the airline? #Person1#: Yes, I did. But I couldn't communicate with them in English. They speak only Spanish. So I need your help. #Person2#: Certainly, sir. What is the flight number and when are you leaving? #Person1#: We are taking IB 385 to London tomorrow at 1 p. m. #Person2#: Oh, I see, sir. We have the airline office inside the hotel. They have an English...	<pad> #Person1# wanted to confirm their flight to London but cannot communicate with the airline in English. #Person2# helps #Person1# to dial 35.</s>	<pad> As for the Spanish speaking staff, #Person1# is configuring the flight to London. The airline has an English-speaking staff and #Person2# offers #Person1# an answer, but #Person1# can't find an answer.</s>	1.717851	1.679462	-0.038389
11	Summarize the following conversation. #Person1#: How much are you asking for this? #Person2#: I'm offering them to you at 150 yuan a piece. Is that all right? #Person1#: Is tax already included in their price? #Person2#: Yes. Our price can't be matched. #Person1#: Would you consider a volume discount? #Person2#: If you buy 1, 000 or more, you'll get a 10 % discount. #Person1#: I'll accept your offer. Summary: </s>	<pad> #Person2# advertises the products @ 150 yuan a piece and offers a volume discount if *100% of purchases is made.</s>	<pad> #Person2# is offering 200 pounds of chip for 150 yuan for 150 yuan a piece. #Person1# will accept #Person2#'s offer.</s>	2.692869	2.545553	-0.147316
12	Summarize the following conversation. #Person1#: Here is the final draft of our contract. I'm glad that we have reached an agreement on almost every term in our trade. #Person2#: Yes, it seems to me we have come quite a long way. However, let me take a close look at the final draft. #Person1#: Do you have some points to bring up? #Person2#: Well, everything we've discussed seems to be here. #Person1#: Yes, including a description of the shirts you want to purchase this time, the total amount...	<pad> #Person1# was happy with two companies' agreement in the final draft of their contract and asked #Person2# to take a quick look at the final draft. #Person2# will read all the stuff that is said by #Person1# regarding each detail but the quality standard is the only standard for their company as it's complete.</s>	<pad> #Person1# is discussing the final draft of the contract of their company. #Person1# mentions the details of terms including the price of each piece, the mode of payment, shipping time, insurance and compensation, and the quality standard. To Sign the Contract, #Person1# wants #Person2# to check over #Person1#'s notes again.</s>	3.201838	3.049746	-0.152092
13	Summarize the following conversation. #Person1#: Could you help me, Sir? My flight got in 15 minutes ago. Everyone else has picked up the luggage but mine hasn't come through. #Person2#: I'm sorry, Madam, I'll go and find out if there is any more to come. Summary: </s>	<pad> #Person1#'s flight got in 15 minutes ago but the baggage is not coming in. #Person2# tries to help #Person1# but #Person2# couldn't help.</s>	<pad> #Person1#'s flight did not come in yet but #Person2# is about to run during #Person1#'s flight.</s>	2.602979	2.394756	-0.208223
14	Summarize the following conversation. #Person1#: Judy, what is everybody talking about? #Person2#: Haven't you heard? Richard was fired by our manager. #Person1#: You're kidding. It can't be true. #Person2#: Believe it or not. Everybody is talking about it in the company. #Person1#: Really? I'm surprised. #Person2#: Me too. Summary: </s>	<pad> Judy and Judy are surprised while they talk about Richard be fired.</s>	<pad> Judy thinks Richard was fired and she's surprised.</s>	1.464138	1.245481	-0.218656
15	Summarize the following conversation. #Person1#: Could you help me figure out how to look for a job? #Person2#: We have lots of options, what type of job do you need? #Person1#: I want to work in an office. #Person2#: Do you want to work part-time or full-time? #Person1#: I want to work full-time. #Person2#: We have binders with local job listings or you can make use of the computers. OK? #Person1#: I am confused a bit but I am sure that I can figure it out. #Person2#: If you make an appoint...	<pad> #Person1# needs to find a job in an office. #Person2# helps #Person1# find job-specific information and tells #Person1# #Person1# can still work for an office. #Person2# also recommends a counselor to #Person1# because #Person1# is not sure.</s>	<pad> #Person1# wants to work full-time in an office. #Person2# tells #Person1# the options available and recommends a counseling service for individual to help choose a job.</s>	2.342860	2.117730	-0.225131
16	Summarize the following conversation. #Person1#: Where shall I register, please? #Person2#: Here. Do you have a registration	<pad> #Person1# wants to register with #Person2# and receive a medical record from	<pad> #Person1# requests #Person2# to register for a medical record and the consultant approves	1.823210	1.587828	-0.235382

	query	response_before	response_after	reward_before	reward_after	reward_diff
	card? #Person1#: Yes. Here you are. #Person2#: Please register your information here and pay for it. And I'll make a medical record for you. #Person1#: OK. How much do I need to pay for the registration? #Person2#: Please pay ten yuan for the registration. #Person1#: Here is my money. #Person2#: This is your registration card. Please don't lose it and bring it whenever...	#Person2#. #Person2# offers: <unk> <unk> <unk> ten yuan to register the information, and what to bring when you're going. Then #Person2# tells #Person1# how to get to the consulting room.</s>	#Person1#'s registration with their payment. #Person2# gives #Person1# instructions on how to get to the doctor's room.</s>			
17	Summarize the following conversation. #Person1#: Oh, my God! What's this? #Person2#: What? #Person1#: Look! This window is open. #Person2#: Did you open it before we left? #Person1#: Are you kidding? It's winter. Why would I open it? #Person2#: I don't know. Wait. Is this yours? #Person1#: No! Oh, my God! Someone has broken into the house. #Person2#: It looks that way. That's probably why the door wasn't locked when we came in. #Person1#: I locked it when I left though. #Person2#: Yes, but t...	<pad> Allen says weird things happened to the window when the house was broken into by a robber. The robber left through the door and locked the windows again which resulted in trying to crack an open window and unlocking. Allen is worried because the robber knows what he stole.</s>	<pad> Allen opens the window but the robber stole the book. He can't find any entry in the house exited by someone who left through the door.</s>	2.378567	2.078370	-0.300198
18	Summarize the following conversation. #Person1#: Hello? #Person2#: Hello? #Person1#: Can I speak to Li Hong, please? #Person2#: Speaking. #Person1#: Hi, Li Hong. This is Alice. #Person2#: Hi, Alice. How are you? #Person1#: Not bad. Li Hong, I am sorry that I can't go to see Mrs. Brown with you tomorrow morning. My mother is ill. I must take care of her. #Person2#: I'm sorry to hear that. You'd better stay at home. After all, we can visit Mrs. Brown later #Person1#: OK. Bye - bye. #Person2#: ...	<pad> Alice cannot go to see Mrs. Brown because her mother is ill and she has to take care of her mother. Li Hong gives her help and she can visit Mrs. Brown later.</s>	<pad> Li Hong lets Alice stay at home and Li Hong reminds her that Mrs. Brown is badly.</s>	1.858839	1.441245	-0.417594
19	Summarize the following conversation. #Person1#: So how did you like the restaurant? #Person2#: Actually, it could have been better. #Person1#: What didn't you like about it? #Person2#: It is a new restaurant. I don't think they have their act together yet. #Person1#: What did you think about the food? #Person2#: I felt that the food was pretty mediocre. #Person1#: The service wasn't that great, either. #Person2#: I agree. The service was not good. #Person1#: Do you think that you want to tr...	<pad> #Person1# asks #Person2# how the restaurant was. #Person2#'s not satisfied with the service and food.</s>	<pad> #Person2# hates the restaurant because of the mess of the food, service and charging fees. #Person2# thinks they have to return once again.</s>	2.239119	1.781267	-0.457852

Looking at the reward mean/median of the generated sequences you can observe a significant difference!