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.

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 [2]:
```
%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/python3.7/site-packages (23.2.1)
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number. pip 23.3 wil
l enforce this behaviour change. A possible replacement is to upgrade to a newer vers
ion of pyodbc or contact the author to suggest that they release a version with a con
forming version number. Discussion can be found at https://github.com/pypa/pip/issue
s/12063
WARNING: Running pip as the 'root' user can result in broken permissions and conflict
ing behaviour with the system package manager. It is recommended to use a virtual env
ironment instead: https://pip.pypa.io/warnings/venv
Note: you may need to restart the kernel to use updated packages.
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number. pip 23.3 wil
l enforce this behaviour change. A possible replacement is to upgrade to a newer vers
ion of pyodbc or contact the author to suggest that they release a version with a con
forming version number. Discussion can be found at https://github.com/pypa/pip/issue
s/12063
WARNING: Running pip as the 'root' user can result in broken permissions and conflict
ing behaviour with the system package manager. It is recommended to use a virtual env
ironment instead: https://pip.pypa.io/warnings/venv
Note: you may need to restart the kernel to use updated packages.
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number. pip 23.3 wil
l enforce this behaviour change. A possible replacement is to upgrade to a newer vers
ion of pyodbc or contact the author to suggest that they release a version with a con
forming version number. Discussion can be found at https://github.com/pypa/pip/issue
s/12063
ERROR: pip's dependency resolver does not currently take into account all the package
s that are installed. This behaviour is the source of the following dependency confli
cts.
pytest-astropy 0.8.0 requires pytest-cov>=2.0, which is not installed.
pytest-astropy 0.8.0 requires pytest-filter-subpackage>=0.1, which is not installed.
spyder 4.0.1 requires pyqt5<5.13; python_version >= "3", which is not installed.
spyder 4.0.1 requires pyqtwebengine<5.13; python_version >= "3", which is not install
ed.
notebook 6.5.5 requires pyzmq<25,>=17, but you have pyzmq 25.1.0 which is incompatibl
e.
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.
sparkmagic 0.20.4 requires nest-asyncio==1.5.5, but you have nest-asyncio 1.5.7 which
is incompatible.
spyder 4.0.1 requires jedi==0.14.1, but you have jedi 0.18.2 which is incompatible.
WARNING: Running pip as the 'root' user can result in broken permissions and conflict
ing behaviour with the system package manager. It is recommended to use a virtual env
ironment 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-bu
ild-c74u7m5q
  Running command git clone --filter=blob:none --quiet https://github.com/lvwerra/tr
l.git /tmp/pip-req-build-c74u7m5q
  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/python3.7/site-packages
(from trl==0.4.2.dev0) (1.13.1)
Requirement already satisfied: transformers>=4.18.0 in /opt/conda/lib/python3.7/site-
packages (from trl==0.4.2.dev0) (4.27.2)
Requirement already satisfied: numpy>=1.18.2 in /opt/conda/lib/python3.7/site-package
s (from trl==0.4.2.dev0) (1.21.6)
Requirement already satisfied: accelerate in /opt/conda/lib/python3.7/site-packages
```

```
(from trl==0.4.2.dev0) (0.20.3)
Requirement already satisfied: datasets in /opt/conda/lib/python3.7/site-packages (fr
om trl==0.4.2.dev0) (2.11.0)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.7/site-pac
kages (from torch>=1.4.0->trl==0.4.2.dev0) (4.7.1)
Requirement already satisfied: nvidia-cuda-runtime-cu11==11.7.99 in /opt/conda/lib/py
thon3.7/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.
7/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/pytho
n3.7/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/pyth
on3.7/site-packages (from torch>=1.4.0->trl==0.4.2.dev0) (11.7.99)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.7/site-packages
(from nvidia-cublas-cu11==11.10.3.66->torch>=1.4.0->trl==0.4.2.dev0) (65.5.1)
Requirement already satisfied: wheel in /opt/conda/lib/python3.7/site-packages (from
nvidia-cublas-cu11==11.10.3.66->torch>=1.4.0->trl==0.4.2.dev0) (0.41.0)
Requirement already satisfied: filelock in /opt/conda/lib/python3.7/site-packages (fr
om transformers>=4.18.0->trl==0.4.2.dev0) (3.0.12)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in /opt/conda/lib/python
3.7/site-packages (from transformers>=4.18.0->trl==0.4.2.dev0) (0.16.4)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site-packa
ges (from transformers>=4.18.0->trl==0.4.2.dev0) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /opt/conda/lib/python3.7/site-packages
(from transformers>=4.18.0->trl==0.4.2.dev0) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /opt/conda/lib/python3.7/site-pac
kages (from transformers>=4.18.0->trl==0.4.2.dev0) (2023.6.3)
Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-packages (fr
om 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/py
thon3.7/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/python3.7/site-packages
(from transformers>=4.18.0->trl==0.4.2.dev0) (4.66.1)
Requirement already satisfied: importlib-metadata in /opt/conda/lib/python3.7/site-pa
ckages (from transformers>=4.18.0->trl==0.4.2.dev0) (6.7.0)
Requirement already satisfied: psutil in /opt/conda/lib/python3.7/site-packages (from
accelerate->trl==0.4.2.dev0) (5.6.7)
Requirement already satisfied: pyarrow>=8.0.0 in /opt/conda/lib/python3.7/site-packag
es (from datasets->trl==0.4.2.dev0) (12.0.1)
Requirement already satisfied: dill<0.3.7,>=0.3.0 in /opt/conda/lib/python3.7/site-pa
ckages (from datasets->trl==0.4.2.dev0) (0.3.6)
Requirement already satisfied: pandas in /opt/conda/lib/python3.7/site-packages (from
datasets->trl==0.4.2.dev0) (1.3.5)
Requirement already satisfied: xxhash in /opt/conda/lib/python3.7/site-packages (from
datasets->trl==0.4.2.dev0) (3.3.0)
Requirement already satisfied: multiprocess in /opt/conda/lib/python3.7/site-packages
(from datasets->trl==0.4.2.dev0) (0.70.14)
Requirement already satisfied: fsspec[http]>=2021.11.1 in /opt/conda/lib/python3.7/si
te-packages (from datasets->trl==0.4.2.dev0) (2023.1.0)
Requirement already satisfied: aiohttp in /opt/conda/lib/python3.7/site-packages (fro
m datasets->trl==0.4.2.dev0) (3.8.5)
Requirement already satisfied: responses<0.19 in /opt/conda/lib/python3.7/site-packag
es (from datasets->trl==0.4.2.dev0) (0.18.0)
Requirement already satisfied: attrs>=17.3.0 in /opt/conda/lib/python3.7/site-package
s (from aiohttp->datasets->trl==0.4.2.dev0) (23.1.0)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /opt/conda/lib/python
3.7/site-packages (from aiohttp->datasets->trl==0.4.2.dev0) (2.0.4)
Requirement already satisfied: multidict<7.0,>=4.5 in /opt/conda/lib/python3.7/site-p
ackages (from aiohttp->datasets->trl==0.4.2.dev0) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /opt/conda/lib/python3.
```

```
7/site-packages (from aiohttp->datasets->trl==0.4.2.dev0) (4.0.2)
Requirement already satisfied: yarl<2.0,>=1.0 in /opt/conda/lib/python3.7/site-packag
es (from aiohttp->datasets->trl==0.4.2.dev0) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /opt/conda/lib/python3.7/site-pac
kages (from aiohttp->datasets->trl==0.4.2.dev0) (1.3.3)
Requirement already satisfied: aiosignal>=1.1.2 in /opt/conda/lib/python3.7/site-pack
ages (from aiohttp->datasets->trl==0.4.2.dev0) (1.3.1)
Requirement already satisfied: asynctest==0.13.0 in /opt/conda/lib/python3.7/site-pac
kages (from aiohttp->datasets->trl==0.4.2.dev0) (0.13.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-packages
(from requests->transformers>=4.18.0->trl==0.4.2.dev0) (2.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.7/site-pa
ckages (from requests->transformers>=4.18.0->trl==0.4.2.dev0) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-pa
ckages (from requests->transformers>=4.18.0->trl==0.4.2.dev0) (2023.7.22)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-packages (f
rom importlib-metadata->transformers>=4.18.0->trl==0.4.2.dev0) (2.2.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/sit
e-packages (from pandas->datasets->trl==0.4.2.dev0) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages
(from pandas->datasets->trl==0.4.2.dev0) (2019.3)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (fr
om python-dateutil>=2.7.3->pandas->datasets->trl==0.4.2.dev0) (1.14.0)
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=67534 sha256=1
6c12d668c7b505961992cff0611a25e5936ec973cc2a35ab3b7302a21a4b2d1
  Stored in directory: /tmp/pip-ephem-wheel-cache-n9nslowe/wheels/41/26/75/08a45cee1a
1bba06c4f340451483cdfe150f4c8dad3876fb2e
Successfully built trl
DEPRECATION: pyodbc 4.0.0-unsupported has a non-standard version number. pip 23.3 wil
l enforce this behaviour change. A possible replacement is to upgrade to a newer vers
ion of pyodbc or contact the author to suggest that they release a version with a con
forming version number. Discussion can be found at https://github.com/pypa/pip/issue
s/12063
Installing collected packages: trl
Successfully installed trl-0.4.2.dev0
WARNING: Running pip as the 'root' user can result in broken permissions and conflict
ing behaviour with the system package manager. It is recommended to use a virtual env
ironment 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 [3]:
```
from transformers import pipeline, AutoTokenizer, AutoModelForSequenceClassification,
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 [4]:

```
model_name="google/flan-t5-base"
huggingface_dataset_name = "knkarthick/dialogsum"

dataset_original = load_dataset(huggingface_dataset_name)

dataset_original
```

```
Downloading readme:   0%|          | 0.00/4.56k [00:00<?, ?B/s]
Downloading and preparing dataset csv/knkarthick--dialogsum to /root/.cache/huggingfa
ce/datasets/knkarthick___csv/knkarthick--dialogsum-391706c81424fc80/0.0.0/6954658bab3
0a358235fa864b05cf819af0e179325c740e4bc853bcc7ec513e1...
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-391706c81424fc80/0.0.0/6954658bab30a358235fa864b05cf819af0
e179325c740e4bc853bcc7ec513e1. Subsequent calls will reuse this data.
  0%|          | 0/3 [00:00<?, ?it/s]
```

Out[4]:
```
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 [5]:
```
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 contair
    """

    # 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_
    dataset = dataset.filter(lambda x: len(x["dialogue"]) > input_min_text_length and 

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

    def tokenize(sample):

        # Wrap each dialogue with the instruction.
        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

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-391706c81424fc80/0.0.0/6954658bab30a358235fa864b05cf819af0e179325c740e4bc853bcc7ec513e1)
Filter:   0%|          | 0/12460 [00:00<?, ? examples/s]
Downloading (…)okenizer_config.json:   0%|          | 0.00/2.54k [00:00<?, ?B/s]
Downloading spiece.model:   0%|          | 0.00/792k [00:00<?, ?B/s]
Downloading (…)/main/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 [6]:
```
!aws s3 cp --recursive s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint
```

```
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/adapter_con
fig.json to peft-dialogue-summary-checkpoint-from-s3/adapter_config.json
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/special_tok
ens_map.json to peft-dialogue-summary-checkpoint-from-s3/special_tokens_map.json
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/tokenizer_c
onfig.json to peft-dialogue-summary-checkpoint-from-s3/tokenizer_config.json
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/tokenizer.j
son to peft-dialogue-summary-checkpoint-from-s3/tokenizer.json
download: s3://dlai-generative-ai/models/peft-dialogue-summary-checkpoint/adapter_mod
el.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 [7]:
```
!ls -alh ./peft-dialogue-summary-checkpoint-from-s3/adapter_model.bin
```

```
-rw-r--r-- 1 root root 14M May 15 11:18 ./peft-dialogue-summary-checkpoint-from-s3/ad
apter_model.bin
```

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

In [8]:
```
def print_number_of_trainable_model_parameters(model):
    trainable_model_params = 0
    all_model_params = 0
    for _, param in model.named_parameters():
        all_model_params += param.numel()
        if param.requires_grad:
            trainable_model_params += param.numel()
    return f"\ntrainable model parameters: {trainable_model_params}\nall model paramet
```

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 [9]:
```
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
)

model = AutoModelForSeq2SeqLM.from_pretrained(model_name,
                                              torch_dtype=torch.bfloat16)

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)

print(f'PEFT model parameters to be updated:\n{print_number_of_trainable_model_paramet
```

```
Downloading (…)lve/main/config.json:   0%|          | 0.00/1.40k [00:00<?, ?B/s]
```

```
Downloading pytorch_model.bin:   0%|          | 0.00/990M [00:00<?, ?B/s]
Downloading (…)neration_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 [10]:
```
ppo_model = AutoModelForSeq2SeqLMWithValueHead.from_pretrained(peft_model,
                                                               torch_dtype=torch.bfloa
                                                               is_trainable=True)

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

```
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 [11]:
```
ref_model = create_reference_model(ppo_model)

print(f'Reference model parameters to be updated:\n{print_number_of_trainable_model_pa
```

```
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 [12]:
```
toxicity_model_name = "facebook/roberta-hate-speech-dynabench-r4-target"
toxicity_tokenizer = AutoTokenizer.from_pretrained(toxicity_model_name, device_map="au
toxicity_model = AutoModelForSequenceClassification.from_pretrained(toxicity_model_nam
print(toxicity_model.config.id2label)
```

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/1.11k [00:00<?, ?B/s]
Downloading (…)olve/main/vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]
Downloading (…)olve/main/merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/239 [00:00<?, ?B/s]
Downloading (…)lve/main/config.json:   0%|          | 0.00/816 [00:00<?, ?B/s]
Downloading pytorch_model.bin:   0%|          | 0.00/499M [00:00<?, ?B/s]
The model weights are not tied. Please use the `tie_weights` method before using the
`infer_auto_device` function.
{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 [13]:
```
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 [14]:
```
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 [15]:
```
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 p
    "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': -2.4896
175861358643}]
[{'label': 'nothate', 'score': 0.9963293671607971}, {'label': 'hate', 'score': 0.0036
70616541057825}]
For toxic text
[{'label': 'hate', 'score': 0.3722729980945587}, {'label': 'nothate', 'score': -0.692
1188831329346}]
[{'label': 'hate', 'score': 0.7435289621353149}, {'label': 'nothate', 'score': 0.2564
7106766700745}]
```

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 [16]:
```
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': -2.4896
175861358643}]
[{'label': 'nothate', 'score': 0.9963293671607971}, {'label': 'hate', 'score': 0.0036
70616541057825}]
```

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

```
[{'label': 'hate', 'score': 0.3722729980945587}, {'label': 'nothate', 'score': -0.692
1188831329346}]
[{'label': 'hate', 'score': 0.7435289621353149}, {'label': 'nothate', 'score': 0.2564
7106766700745}]
```

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 [18]:
```
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 [19]:
```
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 [20]:
```
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=T

        toxicity_score = toxicity_evaluator.compute(predictions=[(input_text + " " + g

        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 [21]:
```
tokenizer = AutoTokenizer.from_pretrained(model_name, device_map="auto")

mean_before_detoxification, std_before_detoxification = evaluate_toxicity(model=ref_mo
                                                                            toxicity_eva
                                                                            tokenizer=to
                                                                            dataset=data
                                                                            num_samples=

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

```
11it [00:25,  2.33s/it]
toxicity [mean, std] before detox: [0.03571996372193098, 0.03741579621043848]
```

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 [22]:
```
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 [23]:
```
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)
```

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 [24]:
```
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`
    reward_tensors = [torch.tensor(reward[not_hate_index]["score"]) for reward in rewa

    # 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 f
ast tokenizer, using the `__call__` method is faster than using a method to encode th
e text followed by a call to the `pad` method to get a padded encoding.
1it [01:42, 102.56s/it]
objective/kl: 29.314075469970703
ppo/returns/mean: -0.5772560238838196
ppo/policy/advantages_mean: 2.246765662405892e-09
-----------------------------------------------------------------------------------
--------------
```

```
2it [03:21, 100.67s/it]
objective/kl: 36.55035400390625
ppo/returns/mean: -0.9952681064605713
ppo/policy/advantages_mean: 4.667979336403505e-09
-------------------------------------------------------------------------------------
--------------
3it [04:51, 95.83s/it]
objective/kl: 26.98063850402832
ppo/returns/mean: -0.5670560002326965
ppo/policy/advantages_mean: 2.5465425324000535e-08
-------------------------------------------------------------------------------------
--------------
4it [06:14, 90.46s/it]
objective/kl: 24.341333389282227
ppo/returns/mean: -0.2767806053161621
ppo/policy/advantages_mean: 1.7039955224618097e-08
-------------------------------------------------------------------------------------
--------------
5it [07:40, 89.02s/it]
objective/kl: 24.389904022216797
ppo/returns/mean: -0.3336139917373657
ppo/policy/advantages_mean: -1.5182209267550206e-08
-------------------------------------------------------------------------------------
--------------
6it [09:20, 92.86s/it]
objective/kl: 27.669343948364258
ppo/returns/mean: -0.5494470596313477
ppo/policy/advantages_mean: 1.347719269517711e-08
-------------------------------------------------------------------------------------
--------------
7it [10:54, 92.99s/it]
objective/kl: 33.35469055175781
ppo/returns/mean: -1.005152940750122
ppo/policy/advantages_mean: 1.670614757642852e-09
-------------------------------------------------------------------------------------
--------------
8it [12:22, 91.55s/it]
objective/kl: 29.939903259277344
ppo/returns/mean: -0.6437324285507202
ppo/policy/advantages_mean: -3.1553877732903857e-08
-------------------------------------------------------------------------------------
--------------
9it [13:56, 92.13s/it]
objective/kl: 30.99266242980957
ppo/returns/mean: -0.7371442317962646
ppo/policy/advantages_mean: -1.1537401256589419e-08
-------------------------------------------------------------------------------------
--------------
10it [15:32, 93.27s/it]
objective/kl: 29.454132080078125
ppo/returns/mean: -0.5535905361175537
ppo/policy/advantages_mean: 5.158094396051638e-09
-------------------------------------------------------------------------------------
--------------
```

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 [25]:
```
mean_after_detoxification, std_after_detoxification = evaluate_toxicity(model=ppo_mode
                                                                         toxicity_evalu
                                                                         tokenizer=toke
                                                                         dataset=datase
                                                                         num_samples=10
print(f'toxicity [mean, std] after detox: [{mean_after_detoxification}, {std_after_det
```

```
11it [00:21,  1.96s/it]
toxicity [mean, std] after detox: [0.02778622367292304, 0.036942100459667684]
```

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

In [26]:
```
mean_improvement = (mean_before_detoxification - mean_after_detoxification) / mean_bef
std_improvement = (std_before_detoxification - std_after_detoxification) / std_before_

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: 22.21%
std: 1.27%
```

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 [27]:
```
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 i
compare_results["response_after"] = [tokenizer.decode(summary_tensors[i]) for i in rar

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

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

```
100%|██████████| 20/20 [01:16<00:00,  3.83s/it]
```

Store and review the results in a DataFrame

In [28]:

```
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_re
df_compare_results_sorted = df_compare_results.sort_values(by=['reward_diff'], ascendi
df_compare_results_sorted
```

Out[28]:

	query	response_before	response_after	reward_before	reward_after	reward_diff
0	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 goes to the company to talk about Richard being fired by their manager. Judy and Judy agree.</s>	<pad> Judy and Judy are surprised that Richard has been fired.</s>	0.815016	2.372849	1.557833
1	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# would like to have a cashed for 10 hundreds and ten twenties in small change and in smaller change when they meet. </s>	<pad> #Person1# asks to have this cashed in small change. Then #Person2# gives #Person1# the name and address of #Person1# and accepts.</s>	1.644222	2.090496	0.446274
2	Summarize the following conversation. #Person1#: Amanda, how do you like this peaked cap?	<pad> Amanda likes the peaked cap but doesn't like the sombrero in black, she thinks it fits her.</s>	<pad> Amanda likes the cap she bought.</s>	0.959776	1.366383	0.406607

	query	response_before	response_after	reward_before	reward_after	reward_diff
	#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>					
3	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> #Person2# doesn't like the restaurant in two ways. #Person2# says the service was not good and has had enough of this place. #Person2# decides to stop.</s>	<pad> #Person2# complained that the service wasn't good and isn't sure if that's forehour. </s>	1.963231	2.311127	0.347896
4	Summarize the following conversation. #Person1#: How much are you asking for this?	<pad> #Person1# decides to buy them for 150 yuan and tells #Person2# they are not matched and	<pad> #Person2# offers the plate to #Person1# at 150 yuan a piece and offers	2.232049	2.565796	0.333747

	query	response_before	response_after	reward_before	reward_after	reward_diff
	#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>	they take a volume discount.</s>	a volume discount till #Person1# buys 1, 000 or more. </s>			
5	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.	<pad> #Person1# mentions how people are getting more and more dependent on the web for buying goods through a personal computer. </s>	<pad> #Person1# explains how people can buy goods through their personal computers and how they can do so.</s>	2.620151	2.937109	0.316959

	query	response_before	response_after	reward_before	reward_after	reward_diff
	#Person2#: Can you tell me how it is done? #Person1#: If a cus...					
6	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# wants to confirm #Person1#'s flight to London, but her colleagues couldn't communicate with her because they speak Spanish. #Person2# provides his phone number and suggests ringing 35.</s>	<pad> #Person1# responds to #Person2#'s prompt. #Person1# wants to reconfirm their flight to London. The airline office has English-speaking staff and #Person1# will dial 35.</s>	1.588624	1.843209	0.254586
7	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,	<pad> #Person1# asks #Person2# how to get to the Cross Bakery building by walking in the opposite direction. #Person2# says east is the opposite. #Person1# thinks #Person2# will show #Person1# the way.</s>	<pad> #Person2# shows #Person1# how to get to the Cross Bakery. #Person1# wants to see the way then #Person2# will show #Person1# the way to the Cross Bakery. </s>	2.677906	2.921665	0.243760

	query	response_before	response_after	reward_before	reward_after	reward_diff
	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...					
8	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# wants to buy some dial-up Internet but #Person1# doesn't want to tie up the phone. #Person2# recommends DEL car by the way it doesn't tie up the phone lines.</s>	<pad> #Person1# wants to order some internet by dial-up with #Person2#'s advice.</s>	2.265636	2.360923	0.095287
9	Summarize the following conversation. #Person1#: What	<pad> #Person2# wants to buy a toy car for his son. #Person1# teaches	<pad> #Person2# wants to buy a toy car for her	1.402472	1.385931	-0.016541

	query	response_before	response_after	reward_before	reward_after	reward_diff
	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>	#Person2# it's under 300 and money is cheaper. </s>	son. #Person1# suggests her a one hundred and twenty which's the cheapest. #Person2# agrees and passes it to the general salesman.</s>			
10	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.	<pad> Alice's mother is sick and Alice can't make it to see Mrs. Brown. She wants to remember that she can come later. </s>	<pad> Li Hong informs Alice that Alice can't visit Mrs. Brown tomorrow morning. Her mother is ill so she can't go to the meeting tomorrow.</s>	2.058562	2.039512	-0.019050

	query	response_before	response_after	reward_before	reward_after	reward_diff
	#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#: ...					
11	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# want to form a rock band and we chat about the other members of the band. #Person1# wants to perform and gives her some observations on staff while #Person2#'s not quite sure who she's playing. Besides fraction, #Person2# comes with their music talent and invites #Person1# to audition.</s>	<pad> #Person1# is forming a music band. #Person1# teaches some instruments and has the possibility of auditioning the other members of the band. </s>	2.608943	2.583678	-0.025265
12	Summarize the following conversation. #Person1#: Here is the final draft of our contract. I'm glad that we have reached an agreement on	<pad> #Person1# has approved the final draft of the contract. #Person2# will check over their notes on every detail now to see what the quality of	<pad> #Person1# and #Person2# want to sign the contract.</s>	3.043200	2.976271	-0.066930

	query	response_before	response_after	reward_before	reward_after	reward_diff
	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...	the agreement is. </s>				
13	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	<pad> #Person1# wants full-time jobs without having to pay a job counselor. #Person2# provides the answers to #Person1#'s questions. #Person2# tells #Person1# there is a job center to help #Person1# find the job and can help #Person1#.</s>	<pad> #Person1# needs to work in an office full-time. #Person2# helps #Person1# understand the topic of looking for a job and helps #Person1# to find a counseling.</s>	2.215091	2.079726	-0.135365

	query	response_before	response_after	reward_before	reward_after	reward_diff
	it out. #Person2#: If you make an appoint...					
14	Summarize the following conversation. #Person1#: Where shall I register, please? #Person2#: Here. Do you have a registration 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...	<pad> #Person2# registers #Person1# for the registration and informed #Person1# how to go to the consultation room. </s>	<pad> #Person1# is going to register for real medical records at the clinic. Numbers are six and the help of #Person2# to help with the registration is given by the staff.</s>	1.668615	1.503923	-0.164692
15	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	<pad> The smoke smells like smoke and #Person2# wants to quit smoking all together. #Person1# recommends, as a different way to quit, the nicotine patch, nicotine chewing gum. A divorce would be perfect for Honey, but he feels exhausted and tired, so he is trying to juggle #Person1#'s urge	<pad> #Person1# tells honey that it smells like ashtray. #Person1# says it needs to be seen some other ways of quitting smoking but could not take it because she can't just quit as she wants a divorce.</s>	1.619366	1.361075	-0.258290

	query	response_before	response_after	reward_before	reward_after	reward_diff
	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...	to reach for his pack of smokes.</s>				
16	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> #Person1# wants to take a coffee break but #Person2# refers to it to stay on the computer. Trevor recommends a coffee break.</s>	<pad> #Person1# wants to take a coffee break but #Person2# can't, so #Person2# has to finish exchanging a report.</s>	2.165697	1.835872	-0.329825
17	Summarize the following conversation. #Person1#: Could you help me, Sir? My flight got in 15	<pad> #Person1# didn't know that everyone else has picked up the luggage but their flight got into 15 minutes ago.</s>	<pad> Mom asks Joe to help #Person1# when her flight comes in. He asked a customer how much luggage	2.286480	1.754310	-0.532170

	query	response_before	response_after	reward_before	reward_after	reward_diff
	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>		she's the one holding so she'll come.</s>			
18	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 tells #Person1# there's someone broke into the house by wanting to be locked up and no one can come in so Allen opens up the window to secure it. They will check upstairs and get some things.</s>	<pad> Allen understands the window was open, so he had to do what he stole. Allen tells #Person1# he robbed the house when he left the house but he didn't find someone. </s>	2.175678	1.545740	-0.629938
19	Summarize the following conversation. #Person1#: Mom, I just finished my paper. Can you proofread it	<pad> #Person1# is proofreading his paper ahead of giving it in. #Person2# appreciates his work on it and tells	<pad> #Person1# looks at #Person1#'s paper and tells #Person1# the original essay will not be lent	2.825037	2.153503	-0.671535

query	response_before	response_after	reward_before	reward_after	reward_diff
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>	him it is worth all the time.</s>	to her teacher. </s>			

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