Making YouTube Titles More Clickbait

By: Jacob DeMuth

Goal: Make YouTube Video Titles "more appealing" to viewers

Subgoal: Make Calix's Video Titles Better

Note: I used GPT2-Large which is the 774M parameter of GPT 2

Install Libraries

In [3]:
```
## !pip install transformers
## !pip install wandb
## !pip install trl
## !pip install pandas
## !pip install datasets
## !pip install accelerate
## !pip install tyro
## !pip install nltk -U
```

Setup Stuff

In [4]:
```
import torch
from tqdm import tqdm
import pandas as pd
import wandb
import os

tqdm.pandas()

from transformers import pipeline, AutoTokenizer
from datasets import load_dataset

from trl import PPOTrainer, PPOConfig, AutoModelForCausalLMWithValueHead
from trl.core import LengthSampler
```

In [5]:
```
config = PPOConfig(
    model_name    = "openai-community/gpt2-large",
    learning_rate = 1.41e-5,
    ## log_with      = "wandb",
)

sent_kwargs = {
         "return_all_scores": True,
         "function_to_apply": "none",
         "batch_size": 16
}
```

In [6]:
```
wandb.init(mode="disabled")
os.environ['WANDB_DISABLED'] = 'true'
```

Load Video Title dataset

Visualize details of dataset

In [7]:
```
## dataset_name="tonarie/Wayback-Data-Youtube-Homepage-Videos"
```

In [8]:
```
ds = load_dataset("csv", data_files="rlhf.csv", split="train")
```

In [9]:
```
ds
```

Out[9]:
```
Dataset({
    features: ['title'],
    num_rows: 42899
})
```

In [10]:
```
ds[15:18]
```

Out[10]:
```
{'title': ['SF International Film Festival Trailer',
  'Re-Enactment: Uncle Buck',
  'Evans Blue "Cold (But I\'m Still Here)" Music Video']}
```

In [11]:
```
from datasets import ClassLabel
import random
import pandas as pd
from IPython.display import display, HTML
```

In [12]:
```
def show_random_elements(dataset, num_examples=20):
    assert num_examples <= len(dataset), "Can't pick more elements than there are i

    picks = []

    for _ in range( num_examples ):

        pick = random.randint(0, len(dataset)-1)
        while pick in picks:
            pick = random.randint(0, len(dataset)-1)
        picks.append(pick)
```

```
    df = pd.DataFrame( dataset[picks] )        ## indexing 10 picks

    print(df)
    print(dataset.features.items())

    for column, typ in dataset.features.items():
        print(column)
        print(typ)
        print(ClassLabel)
        ## The isinstance() function returns True if the specified object
        ## is of the specified type, otherwise False
        if isinstance(typ, ClassLabel):
            print("Hello")
            df[column] = df[column].transform(lambda i: typ.names[i])
            ## print(typ.names[i])

    display(HTML(df.to_html()))
```

In [13]:
```
show_random_elements(ds)
```

	title
0	The Adventures of Batman and Rob...
1	TEXAS WINSS
2	Angelina Jolie Cat Eye
3	motorcycle stunts
4	A Guy On Jets Side Lines Trips A Dolphin Player
5	UFC 117: Chael Sonnen Welcomes Advers...
6	SMACK/ URL Presents K-SHINE vs TAY ROCK
7	American Chopper Visits Jon & Kate Pl...
8	atchoum
9	"The Soul of New Orleans"
10	What Defines a Community?
11	Chris Colfer for The Trevor Project - It Gets ...
12	Eenie Meenie Bikini (Eenie Meenie by Justin B...
13	New GEICO Commercial - Piggy
14	Road Trip
15	How much weight to lift?
16	Tea Party PSA (FreedomWorks) from D.C. Douglas
17	AT&T Don't Text While Driving Documentary
18	George Clooney: Endgame in Sudan
19	Nexus S Review: Best Android Smartphone?

```
dict_items([('title', Value(dtype='string', id=None))])
title
Value(dtype='string', id=None)
<class 'datasets.features.features.ClassLabel'>
```

	title
0	The Adventures of Batman and Rob...
1	TEXAS WINSS
2	Angelina Jolie Cat Eye
3	motorcycle stunts
4	A Guy On Jets Side Lines Trips A Dolphin Player
5	UFC 117: Chael Sonnen Welcomes Advers...
6	SMACK/ URL Presents K-SHINE vs TAY ROCK
7	American Chopper Visits Jon & Kate Pl...
8	atchoum
9	"The Soul of New Orleans"
10	What Defines a Community?
11	Chris Colfer for The Trevor Project - It Gets B...
12	Eenie Meenie Bikini (Eenie Meenie by Justin Bieber Parody)
13	New GEICO Commercial - Piggy
14	Road Trip
15	How much weight to lift?
16	Tea Party PSA (FreedomWorks) from D.C. Douglas
17	AT&T Don't Text While Driving Documentary
18	George Clooney: Endgame in Sudan
19	Nexus S Review: Best Android Smartphone?

In [14]:
```
tokenizer           = AutoTokenizer.from_pretrained(config.model_name)
tokenizer.pad_token = tokenizer.eos_token
```

In [15]:
```
def tokenize( sample ):
    sample["input_ids"] = tokenizer.encode( sample["title"]    )[: 20]
    sample["query"]     = tokenizer.decode( sample["input_ids"] )
    return sample


ds = ds.map(tokenize, batched=False)
ds.set_format(type="torch")
ds
```

Out[15]:
```
Dataset({
    features: ['title', 'input_ids', 'query'],
    num_rows: 42899
})
```

In [16]:
```
ds[15:18]
```

Out[16]:
```
{'title': ['SF International Film Festival Trailer',
  'Re-Enactment: Uncle Buck',
  'Evans Blue "Cold (But I\'m Still Here)" Music Video'],
 'input_ids': [tensor([20802,  4037, 13741, 11117, 36923]),
  tensor([ 3041,    12,  4834,   529,   434,    25, 23169, 13452]),
  tensor([15200,   504,  4518,   366, 34312,   357,  1537,   314,  1101,  7831,
          3423, 16725,  7849,  7623])],
 'query': ['SF International Film Festival Trailer',
  'Re-Enactment: Uncle Buck',
  'Evans Blue "Cold (But I\'m Still Here)" Music Video']}
```

In [17]:
```
def collator(data):
    return dict((key, [d[key] for d in data]) for key in data[0])
```

Load GPT2

In [18]:
```
model     = AutoModelForCausalLMWithValueHead.from_pretrained(config.model_name)
ref_model = AutoModelForCausalLMWithValueHead.from_pretrained(config.model_name)

tokenizer = AutoTokenizer.from_pretrained(config.model_name)

tokenizer.pad_token = tokenizer.eos_token
```

In [19]:
```
ppo_trainer = PPOTrainer(
                config,
                model,
                ref_model,
                tokenizer,
                dataset=ds,
                data_collator=collator
)
```

Load Reward Function

In [20]:
```
device = ppo_trainer.accelerator.device
device
```

Out[20]:
```
device(type='cuda')
```

In [21]:
```
if ppo_trainer.accelerator.num_processes == 1:
    device = 0 if torch.cuda.is_available() else "cpu"  # to avoid a `pipeline` bug

device
```

Out[21]:
```
0
```

In [22]:
```
sentiment_pipe = pipeline("sentiment-analysis", model="christinacdl/XLM_RoBERTa-Cli
```

In [23]:
```
text = "5 Things You Won't Believe"

sentiment_pipe(text, **sent_kwargs)
```

```
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipelines\text_classification.py:105: UserWarning: `return_all_scores` is now deprecated,  if want a similar functionality use `top_k=None` instead of `return_all_scores=True` or `top_k=1` instead of `return_all_scores=False`.
  warnings.warn(
```

Out[23]:
```
[[{'label': 'NOT', 'score': -5.232669353485107},
  {'label': 'CLICKBAIT', 'score': 5.39152193069458}]]
```

In [24]:
```
text = "A Dog Video"
sentiment_pipe(text, **sent_kwargs)
```

Out[24]:
```
[[{'label': 'NOT', 'score': 1.463137149810791},
  {'label': 'CLICKBAIT', 'score': -1.969333291053772}]]
```

Generation settings

In [25]:
```
gen_kwargs = {
         "min_length":   -1,
         "top_k":       0.0,
         "top_p":       1.0,
         "do_sample":  True,
         "pad_token_id": tokenizer.eos_token_id
}
```

Optimize model

Training loop

In [26]:
```
## more then 16 and it runs out of memory

output_min_length     = 4
output_max_length     = 8
output_length_sampler = LengthSampler(output_min_length, output_max_length)
```

In [27]:
```
generation_kwargs = {
    "min_length":     -1,
    "top_k":          0.0,
    "top_p":          1.0,
    "do_sample":    True,
    "pad_token_id": tokenizer.eos_token_id,
}
```

In [28]:
```
## ppo_trainer.config.steps = 100    ## 20,000
ppo_trainer.config.steps
```

Out[28]:
```
20000
```

In [29]:
```
for epoch, batch in tqdm(enumerate(ppo_trainer.dataloader)):
    query_tensors = batch["input_ids"]

    print(query_tensors)
    print(len(query_tensors))
    if epoch == 1:
        break
```

```
1it [00:00,  3.50it/s]
```

```
[tensor([ 5308,   357, 16305,  1820,     8,   290, 32151, 12996, 13777,   366,
         2949,  3125,     1], device='cuda:0'), tensor([   1, 8164,   11, 6914,   1
1, 6914,   11, 6914,    1,  416, 2644],
       device='cuda:0'), tensor([   47,    83,   352,   366, 38667, 34198, 17011,
448, 11097,   220,
        3982,    72,   986], device='cuda:0'), tensor([45478,  6378,  1550, 10216,
1395,  5776], device='cuda:0'), tensor([   72, 23410], device='cuda:0'), tensor([
51,  1045, 33504,   717,  3427, 30424,  1067,   292,   986],
       device='cuda:0'), tensor([   42,  1404,    56, 19878, 18276, 33290,  5064, 30
649,  3539,   402,
        4261,  6561,   357,  1845,  1118,     8,  2644], device='cuda:0'), tensor([
34,  1765,    84,  8211,   376,  1722, 15360], device='cuda:0'), tensor([   35,  250
7,  3691,    13,  6183,   532,   383, 18692],
       device='cuda:0'), tensor([   34,  3861,    57,    56, 47200, 10705,     0,
357,  7556,  3768,
        6407,  8362,     8], device='cuda:0'), tensor([26754,  1636,   309,  9409,
1000, 12274,    64, 31215,  2486,   551,
        1288,   986], device='cuda:0'), tensor([10761,  1565,  9428,  2943, 12576,
5984, 16724,  1546, 43960,  1546],
       device='cuda:0'), tensor([ 88, 929], device='cuda:0'), tensor([   34,   330,
10546,   532,    32,   259,   470, 11853, 13111,   286,
         257, 48248], device='cuda:0'), tensor([33229, 42263,   319,  3271, 18121,
805,  7643,    14,    17,   986],
       device='cuda:0'), tensor([   44,  6849, 16990,  1722, 23896,    25, 27775,  2
688,  4225,  3622,
          82,   986], device='cuda:0'), tensor([ 2437,   284,   307,   257, 27431,
46289], device='cuda:0'), tensor([ 2061, 30587,   284,   921,    30,   532,   569, 1
0652,  5449],
       device='cuda:0'), tensor([ 3398,    36,  1546, 30195,  4261, 30373,   370,  1
847, 28882,    11,
        32247, 10943,  5357,   309,  9693,  4663,  4694, 18723, 17615, 10185],
       device='cuda:0'), tensor([    1,    35, 29309,   553,  3819, 10766,    12, 37
567, 20664,   871,
        4062], device='cuda:0'), tensor([ 6187,  2954,   395, 13145, 10776, 31914,
5157,   329,  1881,   337,
         986], device='cuda:0'), tensor([38887, 42192,   532,  9300, 47983], device
='cuda:0'), tensor([12804,  8402,   360, 16811], device='cuda:0'), tensor([   35,  5
893,  4935,   669,   532,  7831,  1108,  1148,   383, 10028],
       device='cuda:0'), tensor([   49, 17307,   338, 28048,   470,  3412,  9993,  4
377,  3549,    13],
       device='cuda:0'), tensor([26766,  5534,   532, 21167,  1770,    13, 14878,
0],
       device='cuda:0'), tensor([47598,   509, 12484,   259,  2817,   363], device
='cuda:0'), tensor([   42,  5910, 16754,    25,  5849,   838, 38116,   422,  3717],
       device='cuda:0'), tensor([    2,  2484,   707,    83, 16286,  2484,   707,
774,   357,  8205,
        35764, 42736,  8653,  6592,  6387,   370,  8130,  1610,   735,    64],
       device='cuda:0'), tensor([   43, 22312,  9131,    53, 10008,   366, 15567,  3
698,     1,   554,
        6462], device='cuda:0'), tensor([ 2437,   284, 25715,  9032,   534, 26915,
1220,  7133,  1312,   986],
       device='cuda:0'), tensor([ 2504, 10358,  1355,  2185,   685, 27245,    60, 10
799, 42263],
       device='cuda:0'), tensor([   33,  2433, 14995, 17381,   857,   257,   736,  2
704,   541,   355,
         257,  4681, 39485,   986], device='cuda:0'), tensor([   49,  4449,  7697,
```

532, 6912, 10117,   13, 6960,   449, 1453,
         7357,   357, 12710,   986], device='cuda:0'), tensor([39163,   532, 21285,
2159, 5454, 3050,   532, 4492, 13614, 23575,
          532, 7769,   14, 3312,   14,   940], device='cuda:0'), tensor([9039,
25285, 20989,   82,   287, 9498,   286, 1355,   986],
       device='cuda:0'), tensor([1, 22911,   1,   422,   366,   464, 26632,
1,   416, 5498,
         7755,   357,   448,   783,   8], device='cuda:0'), tensor([1639, 2185,
1629, 9699,   532, 16160, 2080, 2185,   532, 6328,
           72,   986], device='cuda:0'), tensor([50, 14057, 12205, 4703,   341,
329,   309,   398, 15992,   11,
        15282, 1222, 1345,   403, 1362], device='cuda:0'), tensor([2061, 2896, 11
27, 257, 8108,   30], device='cuda:0'), tensor([34,   735, 13199, 3862,   0],
device='cuda:0'), tensor([14350,   313, 3334, 3961], device='cuda:0'), tensor([
39, 1794,   699, 1222, 14668, 7623, 8315,   263, 24534, 15181,
          986], device='cuda:0'), tensor([47889,   284, 7066, 3020,   868], device
='cuda:0'), tensor([6090, 4403, 12002, 13575,   30], device='cuda:0'), tensor([10
603, 3232, 4993, 16291, 40027, 15869], device='cuda:0'), tensor([25082, 12200, 142
13,   532, 5648,   344, 14438,   257, 5045,   986],
       device='cuda:0'), tensor([33246, 1437, 39329,   494,   220,   287, 25051, 3
050],
       device='cuda:0'), tensor([45272, 6705, 13804,   532, 23432,   284, 3082, 14
471],
       device='cuda:0'), tensor([47, 5036,   487,   364, 1050,   323, 1035, 45
371, 30830, 11752,
          710,   797,   461, 5378,   30,   7,   32,   84, 2704, 9101],
       device='cuda:0'), tensor([41983, 6369, 37997, 1961,   25, 43821,   14,
34, 1847, 5064,
        30649, 3539,   402, 4261, 6561], device='cuda:0'), tensor([44534, 17342,
40244, 6612, 3691, 27620,   324, 32355,   986],
       device='cuda:0'), tensor([40,   56, 22778,   532, 23635, 17546,   357,
41, 3861,   11,
        25347, 31918,   11, 2644], device='cuda:0'), tensor([464, 6305, 3653,
532, 10078,   338, 15875, 6602],
       device='cuda:0'), tensor([10970, 17306, 4146, 9947,   311, 4760,   32,
25, 12682, 13070,
         1340,   532, 1475, 2527,   72,   986], device='cuda:0'), tensor([39,
10339,   351, 14536], device='cuda:0'), tensor([45, 363, 4948, 1689,   25, 317,
6407, 8362], device='cuda:0'), tensor([25082, 16032,   422, 18622, 11176,   574,   5
32, 3648],
       device='cuda:0'), tensor([18601, 3843, 14603, 24994,   0], device='cuda:
0'), tensor([17584, 6064, 4103,   25, 34753, 9957, 6289], device='cuda:0'), ten
sor([1314,   400, 13641, 9570], device='cuda:0'), tensor([2662,   362,   532,   6
04, 6599,   38,   357,   51, 20958, 10287,
          475,   82, 5145,   8], device='cuda:0'), tensor([6,   464, 48623,
2977,   6, 36923, 5572], device='cuda:0'), tensor([7376,   69, 21663, 9677, 4
146], device='cuda:0'), tensor([24750,   448,   25,   968, 9621, 3811, 9579, 429
75,   13, 1485,
            7, 16371, 6912,   14,   986], device='cuda:0'), tensor([33229, 42263,
319, 3271, 18121,   805, 7643,   14,   17,   986],
       device='cuda:0'), tensor([25284, 28218], device='cuda:0'), tensor([12256, 69
26,   338, 5521,   448], device='cuda:0'), tensor([9031, 4029, 45757,   532,   36
6, 7738, 16308, 3683,   286, 1680,
         5075,   1], device='cuda:0'), tensor([12322,   281, 4196], device='cuda:
0'), tensor([36, 1084,   368,   532, 28285], device='cuda:0'), tensor([11473, 11
860,   50], device='cuda:0'), tensor([3791, 22319, 22707, 22724,   532, 23097, 13

60], device='cuda:0'), tensor([45,   13,   56,   13, 1220, 17718,   25, 440
3, 42162,   319,
        25005,   11, 2773, 5930,   912, 14190,   532,   220,   299, 20760],
       device='cuda:0'), tensor([7902,   52, 34958, 1137, 11357, 2488, 14417,
371, 30194, 14114,
         357,   486,   14, 2919,   14, 1157,   8], device='cuda:0'), tensor([
4965, 20080, 11460, 8607,   347, 17456, 36863, 1671,   292,   532,
         449,   986], device='cuda:0'), tensor([2061,   775, 19882,   379,   262,
17551, 11739, 27752,   287, 6257],
       device='cuda:0'), tensor([31080, 1016, 7165,   329, 11226, 1241, 3126,
287,   995,   286,
        1175, 3323], device='cuda:0'), tensor([38,   528, 4666,   78, 20101,
26094, 4793, 22237,   12,   261,
          25, 10397, 1195,   16], device='cuda:0'), tensor([1026, 29620, 11625],
device='cuda:0'), tensor([464, 5506,   726,   276, 35832, 35513,   0], device
='cuda:0'), tensor([50, 17731, 14519,   244, 13585,   498], device='cuda:0'), ten
sor([8134, 1564], device='cuda:0'), tensor([4261,   34, 532, 1578, 4718, 328, 50
7, 5834], device='cuda:0'), tensor([38, 1934,   64,   12, 23830, 33743], devic
e='cuda:0'), tensor([25082, 6479, 13842,   25,   383, 12281], device='cuda:0'), te
nsor([7454, 16055,   532,   317, 22607, 39771,   532, 12458,   574, 2644],
       device='cuda:0'), tensor([12804, 13051,   11, 21939, 13051], device='cuda:
0'), tensor([34, 26505, 28548,   327, 44937, 13727, 12491, 1424],
       device='cuda:0'), tensor([45565,   0, 45565,   0], device='cuda:0'), tens
or([464, 2351,   532, 7547,   422, 12232,   319,   642,   14, 1314,
         357, 15721, 5329,   8,   532,   317,   569,   986],
       device='cuda:0'), tensor([3347,   79, 4176, 30382,   82, 5426, 3000, 5
947, 3226, 20997],
       device='cuda:0'), tensor([22975,   30, 11382, 6521], device='cuda:0'), tens
or([20266, 6047,   532, 6889,   929, 38749], device='cuda:0'), tensor([27977, 214
9, 12576,   968, 6869, 36923], device='cuda:0'), tensor([40,   25, 7922, 1881], d
evice='cuda:0'), tensor([47184], device='cuda:0'), tensor([5124, 3691,   13, 618
3,   532, 9870,   262,   371, 3008,   930,
        13836, 3691,   13, 6183], device='cuda:0'), tensor([27722,   12,   51,
1726,   262, 3000, 1303,   24,   25, 20715,
          13, 1535, 1337,   13, 1578, 7973,   13], device='cuda:0'), tensor([
5005, 26408, 6612, 2908, 10366, 15968, 12139, 19947,   492, 18844,
         986], device='cuda:0'), tensor([53, 32667], device='cuda:0'), tensor([1
2310, 8120,   14, 10289, 47102,   412, 18647, 3691, 6997,   13,
          39,   436,   293], device='cuda:0'), tensor([23808,   338, 10175, 4803,
25, 36483, 1355, 19970,   290,   262,
        14801], device='cuda:0'), tensor([7738, 6964, 5521, 448,   13], device='cud
a:0'), tensor([9861, 1697, 15971,   25, 10816, 11095, 13690, 1424,   327,   98
6],
       device='cuda:0'), tensor([48034, 26729, 7849, 7623], device='cuda:0'), tens
or([35, 19897, 31914, 2688,   532,   352, 12490,   583, 1524],
       device='cuda:0'), tensor([33, 35298, 49341,   11, 8533, 17738], device
='cuda:0'), tensor([36964,   359, 1029, 1524,   338, 14779, 9280,   0],
       device='cuda:0'), tensor([10919,   857, 16592, 25772,   910,   546, 2644], d
evice='cuda:0'), tensor([13300,   921, 10358, 15648,   357, 13256,   290, 1004,
65, 1313,
          8], device='cuda:0'), tensor([26103,   645,   427,   16,   83,   11,
5230, 1842,   523,   986],
       device='cuda:0'), tensor([15948, 41746,   684, 2576,   329, 4814, 1524], d
evice='cuda:0'), tensor([464, 7868,   286, 11526, 19566,   532, 32722, 29620, 350
80,   276],
       device='cuda:0'), tensor([5840, 2850, 7119, 292, 418], device='cuda:0'), te

```
nsor([18960,   642,  4463,  1000, 43626,   532, 13980,   338,  1052,  5549,
          986], device='cuda:0'), tensor([   51,  4163,   284,  3362, 27698], device
='cuda:0'), tensor([12832, 21618, 10047,  2015, 12199,  5261,  8326,   357,   259,
317,
            2,  4159,     8], device='cuda:0'), tensor([ 1120,  4897,  1338,  9437,
364,   287,   604, 23757],
       device='cuda:0'), tensor([ 3629,    78,  8952, 10215,   568,   569,   971,
641, 10247,   446,
         3717,   532,  1024,   569,  2675,    65,   567,  2530,   268],
       device='cuda:0'), tensor([43559,  3180, 29809,  2149, 19807, 25848,   220,
317, 35507,  3963,
        11948,   986], device='cuda:0'), tensor([   33,   853,   391, 23102, 18675],
device='cuda:0'), tensor([12050,  3406, 11744,   350,   382, 26137,   862], device
='cuda:0'), tensor([43227, 13876,   805,  2364,  1058, 40560,  1046,   343,  3367,
920,
         1186,  2013,   288,     6, 24419,   559,  2395,  5145],
       device='cuda:0'), tensor([   42,  2389,   360,  5883,    33, 12473,  3069,
50,     0],
       device='cuda:0'), tensor([44721,  6462, 33998,  5693, 36361], device='cuda:
0'), tensor([ 3123,  1052,   377,   272,   555, 26849,   257, 24568, 10115, 36309,
          390, 15095, 10094,   986], device='cuda:0'), tensor([43421], device='cuda:
0')]
128
[tensor([37411,  1815,  1694, 22724], device='cuda:0'), tensor([ 2484,   461,  8704,
532,   406, 11216, 10117,    13,   360,  6457,
         1453,   371, 27747], device='cuda:0'), tensor([39914, 12558, 22637, 19067,
37299,  2271, 12558, 21167,  2688,  5075],
       device='cuda:0'), tensor([21659,    10,   532,  6139, 23087], device='cuda:
0'), tensor([ 5796,   333,    69, 10359,  1725,   379,  5575, 13822,   260,   357,
        40313,     8], device='cuda:0'), tensor([   34,  5918,  3232, 22481,   416,
20071,  1812,   338, 12670,   346,
        30187,    81,    89,  5173,   988], device='cuda:0'), tensor([  818,  4703,
35967,  1095,   532, 22724,  2547,  1118],
       device='cuda:0'), tensor([ 5990, 15464,  1114,   609,  1942,   344,    88,
532,  7922,   352],
       device='cuda:0'), tensor([10161,  1820, 47595,   357,    37,  1157,  1303,
940,     8],
       device='cuda:0'), tensor([    1, 12443,   341,  5800,  8949,     1,   416,  9
817, 24332],
       device='cuda:0'), tensor([ 3791,  6280, 31104,  4793, 14628,   968,  6280,  4
380,   286,   262,
         2159,     0], device='cuda:0'), tensor([ 3856,   620, 23432, 23102], device
='cuda:0'), tensor([25082, 21494,   532, 13707, 20843,   986], device='cuda:0'), ten
sor([   54, 14887, 48294,  3963,  9834, 34677, 12248], device='cuda:0'), tensor([
55,  5776,  1467,    25,  6416,  3530,   532, 14757,  5648, 49325],
       device='cuda:0'), tensor([12442, 10682,  9252,   362,  6857,  9579,   532,  2
142,   352,   532,
        37219,   986], device='cuda:0'), tensor([   38,   349,  1052,   377,  4533,
257,    78,   327, 12514,    13,
        39702,   551,  8528,  1698,   418], device='cuda:0'), tensor([46667, 14152],
device='cuda:0'), tensor([   51,  1404, 30259,  4522], device='cuda:0'), tensor([
1,  3987,   470, 13707,  3944,   494,  7114, 29653,   532, 11182,
        42192,    11,  2574,  1122,  1757,    11,  8225,  4169,   986],
       device='cuda:0'), tensor([ 3987, 48214,  3754,  9299,  3700,  4373,  1222,
978, 15465, 10972,
         319,  8653, 16835], device='cuda:0'), tensor([   32,  3619, 11563,  6786],
```

```
device='cuda:0'), tensor([ 39, 747], device='cuda:0'), tensor([   36, 43130,  8207,
338,  8362], device='cuda:0'), tensor([15285, 20068,  2394,  3398,    56,  5870,
36,  1137,  2538,  2885,
        4877,     0], device='cuda:0'), tensor([39299,   666,  7542,  6889,   929,
36361,  1343, 25134,   679, 21975,
           0], device='cuda:0'), tensor([38887, 42192,   532,  2080, 23058, 31248,
357, 18947,  2488,   986],
      device='cuda:0'), tensor([1102, 4399, 2832,  582], device='cuda:0'), tensor
([31443,    88, 12579], device='cuda:0'), tensor([47849, 14715,   347,  8228,  3245,
4424,   689,  7444],
      device='cuda:0'), tensor([23303,  9618,   370, 13210,   979,    74,  1220, 47
808, 21717,  1220,
        2644], device='cuda:0'), tensor([ 3791, 12590,  1222,  2851, 37838], device
='cuda:0'), tensor([43719, 10169, 11882,  3691,  4216,  6902,   305,  1294,  4946,
8125,
        3717, 25640, 26532,   986], device='cuda:0'), tensor([21902,   353, 16071,
287,  4687], device='cuda:0'), tensor([   37,  1313,    74,   399,  1024,  2188,   5
32,   383, 34414, 18888],
      device='cuda:0'), tensor([   41, 25218,  3619,  9973,   532, 41249,   317,  4
401],
      device='cuda:0'), tensor([13887,  7897,   287,  2869,  3373], device='cuda:
0'), tensor([   45,   375,   283, 26105,   270,  1077,  8903,  3907,  4528, 13095,
         554,   406,   986], device='cuda:0'), tensor([ 6187,   388,   290, 19829,
23896,   357, 35028,  5961,  1134,     8,
         351,   797,  1130,  3609], device='cuda:0'), tensor([31439, 20745,   362,
15932, 24204], device='cuda:0'), tensor([18844,   805,   532,   383, 16346], device
='cuda:0'), tensor([2601,  420, 5439,   11,  262, 3797], device='cuda:0'), tensor([
3987,   470,  3497,  3406,  4100,  3619,   276,   532,  1312, 36420,
          52], device='cuda:0'), tensor([19206, 19161,     6,  5075, 13863,  1520,
594,   434, 17917],
      device='cuda:0'), tensor([ 3666, 22505,  6084,  2894,  7875, 36844,  3827,  5
143,  2750, 12400,
       13022,   532,  2644], device='cuda:0'), tensor([13256,  6612,   285,   518,
260,   257, 22346,  2026,   257, 12654,
         418], device='cuda:0'), tensor([42997, 25778,   532, 36923], device='cuda:
0'), tensor([30568,  3250,   220, 36458,   259,  1088,    14, 45896, 10823,   357,
        9631,     8], device='cuda:0'), tensor([   39,   538,   265, 11815,   317,
416, 18008,   672, 10339,    11,
        3457], device='cuda:0'), tensor([   49,  2246,  8220,  1340, 26195,  8120,
50,   347,  6242,    56,
           0], device='cuda:0'), tensor([ 3843,  1450,   471, 37321], device='cuda:
0'), tensor([12442, 39139,  9252,   532,  7922,   513], device='cuda:0'), tensor([
16,    14,  1507,    14,  1157,   532, 13407,  2561,  1355, 22607,
        2142,   352], device='cuda:0'), tensor([11187,  5355,  7092], device='cuda:
0'), tensor([   34,   641, 11094,   379, 46757,  3717,   532,  6365,   416, 43376,
         986], device='cuda:0'), tensor([11002, 23058, 31248,  3242,    11, 31973,
5181,     0],
      device='cuda:0'), tensor([37114,   338, 11853, 29505,   338, 14919, 37619,
532,   257,  3298,
       28422, 18098], device='cuda:0'), tensor([20366, 10992, 21276], device='cuda:
0'), tensor([45087, 30007,   741,   259,   319, 13633,  2677,   807,    13,  1495,
          13,  2931,   376,   986], device='cuda:0'), tensor([37181, 12887, 10351,
16289, 44156,  2849,   362], device='cuda:0'), tensor([45272,  6705, 13804,   532,
383,  7491,  1525,  7542],
      device='cuda:0'), tensor([ 464, 3819, 6295], device='cuda:0'), tensor([14662,
287,   257,  3596,  1665,  6005,  1303,    17,    25,  6575],
```

```
       device='cuda:0'), tensor([   32,  9927,  8737,  1720, 25647,     0], device
='cuda:0'), tensor([   36,  1084,   368,   532,  1400,  5896,   357, 27594,    13, 1
6342,
         370, 23495,     8,   357,  8610,    48,     8], device='cuda:0'), tensor
([32782, 30709,  5883,  5550, 32761,  2849,     0], device='cuda:0'), tensor([ 6187,
13, 18024, 47595,   406,  2752, 14331], device='cuda:0'), tensor([37136,   262, 4273
9, 22777, 23097], device='cuda:0'), tensor([19585, 46838,    25,  8559,   287,   25
7,  1086,  2194],
       device='cuda:0'), tensor([ 7707,  2390,  1404, 22707, 15859, 25256,    36, 30
774,   337,  6369,
        22707, 17368,   986], device='cuda:0'), tensor([   40,   509, 41841,    13,
314,   509, 41841,    13,   796,     8],
       device='cuda:0'), tensor([22125,  7336, 11895,  2394, 45463, 44423,  3336,
376,  3843, 11335,
         532, 34958,   986], device='cuda:0'), tensor([   52,  6080,  8756, 36242,
287,  9643,  1220,  7130,   912, 14597,
        5145,  5145,  5145], device='cuda:0'), tensor([38887, 42192,   319,  3909,
5265,  7547,  2556,  2644],
       device='cuda:0'), tensor([10798,   286, 49227, 15976, 36577, 19098], device
='cuda:0'), tensor([ 3987,   470,   575,   962,  2011, 21853,   357,  7841,   352,
8],
       device='cuda:0'), tensor([14967, 27209,   338,   860,   383,   265,  8143, 36
923,  5572,  2644],
       device='cuda:0'), tensor([32942,  5809,  1158,  1879,  3228], device='cuda:
0'), tensor([36462, 11891], device='cuda:0'), tensor([   50, 10546, 12898,  4373,  6
251,   634,   284,  2031,  1377, 28537,
          11,  2644], device='cuda:0'), tensor([14772, 23532, 39708,  1550,  5338,
370,  1187,  1675,  1355,   317,
        13756,   986], device='cuda:0'), tensor([   44,  9618, 34305,  3615,   287,
262,   471,    13,    50,    13,
          32,    13,   532, 20787,   262, 49525], device='cuda:0'), tensor([11922,
12287, 12533, 13696, 14954,   379,   309,   986],
       device='cuda:0'), tensor([16775,    25, 19100,  3050,    25, 17905,   422,
262, 42754],
       device='cuda:0'), tensor([ 3118,  2375,   376,  7834, 35900,  7298,  1766,
276,    12,    88,
          12,    33,   918,   259], device='cuda:0'), tensor([1544,  466], device
='cuda:0'), tensor([   35,  2507,  3691,    13,  6183,   532,   383, 18692],
       device='cuda:0'), tensor([43887, 19699, 11809, 16409,   284, 23811], device
='cuda:0'), tensor([   43, 16696,    56, 36871, 13246,   311,  2538,    53,  1268, 3
6923],
       device='cuda:0'), tensor([   23,    12, 26094, 21827,    75,   485,   287, 32
744, 36821,     0],
       device='cuda:0'), tensor([ 464, 5184, 5375, 8609], device='cuda:0'), tensor
([7109, 2135, 3364,  616, 2460, 1057], device='cuda:0'), tensor([   45,  1268, 3704
8,  9348,  1503,  3228], device='cuda:0'), tensor([ 4221,    49, 32632,   317, 1018
8, 10705, 46455, 11319, 12268],
       device='cuda:0'), tensor([   39,  6724,  3619,   805, 26049,  2097, 25756,
532,   383,   767,
        4426,  1041,    73,   721,   986], device='cuda:0'), tensor([28886, 21259,
4848,   274,  2619,  1869, 34872,  5722,  3226, 14732,
        8307], device='cuda:0'), tensor([   45,  5733, 10185], device='cuda:0'), te
nsor([43471,  2414,    25,  5833,  2768,  5411], device='cuda:0'), tensor([    1,  5
886,   262, 27272,     1,   532,   703,   257,  6588,  1451,
        2499,   532,   416, 31232,  7567,  5702], device='cuda:0'), tensor([15633,
14708,  3691, 10029, 15142,   362,    12,    21,   685,    17,
```

```
          14,  2713,   986], device='cuda:0'), tensor([ 8021, 44961,   338, 24299,
17987, 39759,   955, 12061,  6857,  9579,
         685, 21991,   986], device='cuda:0'), tensor([   44,  5673,  7102, 48842,
1961, 23255, 28206, 13315,  5870, 16696,
        24653,   986], device='cuda:0'), tensor([20148, 12136, 14801,  3228], device
='cuda:0'), tensor([ 2704,   702,   287,   479,    79,    74,   220,   220, 12612,
321,
         279,   461,  4103,    13,  2704,    85], device='cuda:0'), tensor([24510,
2879, 21869,    25,   383,  6416, 21561,    25,  9849, 21869,
         287,   257,  5752,     0,   383, 13745], device='cuda:0'), tensor([27369,
7623,    25,  5524,  9666,  2879,   274, 12243,  2159, 13266],
       device='cuda:0'), tensor([5703, 1881, 3125, 6914], device='cuda:0'), tensor([
54, 14887, 48294,  3963, 16400,  5959, 11319,  9297, 11598, 12429,
        4303,  6684,    37,  1174], device='cuda:0'), tensor([ 7762,   298,  2879,
2185,  1039,   262, 24165,  7246,  2935,    76,
        1335, 44070, 14714,  1157,   532,   604,    14,    21],
       device='cuda:0'), tensor([14134,   286, 18104,    25, 12495, 20745,   362,
532, 40713,   986],
       device='cuda:0'), tensor([  39,  648,  259,  262, 7825,  444], device='cuda:
0'), tensor([12805,  4061,  4935,  2297, 25379,   532, 16032,  5555, 13879],
       device='cuda:0'), tensor([ 3198, 35708,    25,  1881, 23425, 40440,   284,  2
644],
       device='cuda:0'), tensor([    1, 12442,  3431,     1, 17372,   357,  3673,  2
014,   532,   277,
         986], device='cuda:0'), tensor([31077,   284,  1992, 36468,   261], device
='cuda:0'), tensor([ 2200,    25,   311,  6369,    56, 28196,   422, 22814,    51,
9936,
         371, 11860,    56, 29620,   366, 25621,  3705,    56,     1,  3228],
       device='cuda:0'), tensor([ 1120,  1979,   338,  7413,   314, 12189,  8145,  1
356,   357,  6767,
        6005,     8], device='cuda:0'), tensor([   42,  3325, 21681,   569,   324,
23267,   350,   385,    84,  6135,
          13,   347,  9101,    75,  9116,    76,   807,    13,   509, 30102],
       device='cuda:0'), tensor([48890,   376,  7626,  3648,    25,   371,    17,
12,    35,    17,
        10934,   364], device='cuda:0'), tensor([  464, 32011, 36238], device='cuda:
0'), tensor([43413,   620, 42032,  3594,  3834,  2142,   362,    14,    17],
       device='cuda:0'), tensor([28219,  4380,   532,  7922,   352], device='cuda:
0'), tensor([14134,   286, 18104,    25, 12495, 20745,   362,   532, 18291, 41472,
         532,   370, 11402,   496], device='cuda:0'), tensor([ 2484,   265,  1008,
338,  2907, 12338, 33957], device='cuda:0'), tensor([12543,  3281,   355,  5968], de
vice='cuda:0'), tensor([14618,  1675,   383,  7444,  8741], device='cuda:0'), tensor
([  32, 1416, 3004], device='cuda:0'), tensor([   36,    12,  7762,   298,   500, 34
635, 13896,  3228],
       device='cuda:0')]
128
```

In [30]:

```python
for epoch, batch in tqdm(enumerate(ppo_trainer.dataloader)):
    query_tensors = batch["input_ids"]
    print(epoch)
    print(batch)
    print('**********************')
    print('**********************')
    print('**********************')
    print('**********************')
    #### Get response from gpt2
```

```
    response_tensors = []
    for query in query_tensors:
        gen_len                            = output_length_sampler()
        generation_kwargs["max_new_tokens"] = gen_len
        response                           = ppo_trainer.generate(query, **generat
        response_tensors.append( response.squeeze()[-gen_len:] )
    batch["response"] = [ tokenizer.decode(r.squeeze()) for r in response_tensors ]
    print(batch)
    if epoch == 1:
        break
```

```
0it [00:00, ?it/s]
```

0
{'input_ids': [tensor([31466,   298, 5438, 18448,   13], device='cuda:0'), tensor
([37247,   868,   287, 9626,   370, 24929], device='cuda:0'), tensor([18332,   278,
2253,   350, 4090,   351, 4705, 33572],
       device='cuda:0'), tensor([40555, 2502, 5851, 12378, 1108], device='cuda:
0'), tensor([38, 4754, 5532, 2873, 1755, 9581, 1570,   422, 25762, 4391,
        24073], device='cuda:0'), tensor([2437,   284, 48887, 39801], device='cuda:
0'), tensor([42, 5910, 16754,   843, 13308,   64, 7459, 22244, 1114,   986],
       device='cuda:0'), tensor([42731, 11927,   0], device='cuda:0'), tensor([195
55, 2257, 6833, 2310, 10891,   72, 5472], device='cuda:0'), tensor([15645, 437
3, 2218,   13, 7039, 4908,   532, 11806,   64, 1629,
         2185], device='cuda:0'), tensor([54, 7697, 11763,   317, 9897,   30],
device='cuda:0'), tensor([24129, 2611, 32189,   513,   14,   18], device='cuda:
0'), tensor([464, 11165, 16089, 13388], device='cuda:0'), tensor([43930,   290, 43
931,   25,   402, 9872, 34352,   13,   362,   357,
         3163,   363, 1211,   8], device='cuda:0'), tensor([20630, 49869, 5411,
8541], device='cuda:0'), tensor([464, 2351,   532, 7547,   422, 12232,   319,
642,   14, 1314,
          357, 15721, 5329,   8,   532,   317,   569,   986],
       device='cuda:0'), tensor([10603,   338,   362,   358,   406,   853,   395, 27
351, 2442, 3776,
          532, 7324,   13, 48797, 38793,   13, 21283], device='cuda:0'), tensor([
18,   67, 11034, 1790,   366,   33,   776, 3862,   1],
       device='cuda:0'), tensor([42, 2737], device='cuda:0'), tensor([37798, 120
3, 6491, 18034, 3331, 29979], device='cuda:0'), tensor([10970, 9348,   32, 595
9,   532, 36923], device='cuda:0'), tensor([37798, 26690,   532, 7232,   296, 1985
1], device='cuda:0'), tensor([25881, 22267, 3854, 27431, 46289, 3811, 4868], devi
ce='cuda:0'), tensor([18927], device='cuda:0'), tensor([42770,   371, 3266, 31225,
3764, 10850], device='cuda:0'), tensor([18453,   25, 6889,   929, 37460, 10897], d
evice='cuda:0'), tensor([8021, 44961,   338, 24299, 2873, 6602], device='cuda:
0'), tensor([49061, 4333, 6910, 3764], device='cuda:0'), tensor([43719, 10169, 11
882, 24205,   877, 14307,   383, 3564,   986],
       device='cuda:0'), tensor([1, 9328, 1677, 11879,   360, 32235,   1,
416, 36715, 2448,
          986], device='cuda:0'), tensor([14967, 11130, 1831,   338, 7444, 3811,
4868], device='cuda:0'), tensor([34, 6765,   73, 27498,   357, 27399, 47756,
8, 1569, 17879,
          292, 1188, 29634], device='cuda:0'), tensor([30365, 27854, 1122,   532,
14801, 11853, 5157,   986],
       device='cuda:0'), tensor([27827,   338, 43528, 27558,   25, 23095, 1303, 3
261],
       device='cuda:0'), tensor([42315,   25, 3776, 3827], device='cuda:0'), tens
or([53, 1000,   84,   532, 1475, 2501,   292, 31842],
       device='cuda:0'), tensor([33869, 33195, 34017, 20014,   532, 3596,   362,
0],
       device='cuda:0'), tensor([7437, 34392, 2907, 7244, 36978, 1629, 1962,
986],
       device='cuda:0'), tensor([42149, 19549, 6489,   56,   11, 1714, 6386, 14
643, 3715, 1683,
          986], device='cuda:0'), tensor([33754,   323, 2271,   383, 16781,   532,
383, 14270],
       device='cuda:0'), tensor([3064,   23, 2481, 7244, 15062, 1154, 20350, 9
712],
       device='cuda:0'), tensor([18454, 6415,   520, 17479,   290, 15335, 5274,
16, 5068],
       device='cuda:0'), tensor([56, 13513, 20165, 31868, 1754,   532, 1114, 11

```
254],
       device='cuda:0'), tensor([   34,  4131,   563,  5341,   371,   620,   805,
259,  2364, 29278,
        33375,  1400,    13,   604], device='cuda:0'), tensor([45467, 30593,     6,
7403, 15120], device='cuda:0'), tensor([40331,  1137,    33,  3913,    43, 35329, 26
442,     0,   357,    17,
          14,    22,    14,   940,    12, 23601,     8], device='cuda:0'), tensor([
5446,  3185, 20151, 14277,  2043, 14990,  1546,   357, 13295, 10628,
           8], device='cuda:0'), tensor([ 2437,   284, 22892,   317,  5181,   532,
2750, 10370,  2332,  1313],
       device='cuda:0'), tensor([   36,  1084,   368,   532,   383, 15932,   357,  1
526,  9520, 31612,
        1222,   986], device='cuda:0'), tensor([  464, 14087,   286, 14426,   266,
14,  6896,    45, 15672,    25,
        2159,   286,  1810,  6098,   986], device='cuda:0'), tensor([    1,  2504,
338,  1867,  6786, 28453,  1675,  2185,     1,   220,
        7849,  7623,  1675, 19899,   986], device='cuda:0'), tensor([  168,   228,
234,   167,   227,   222,   168,   233,   250,   167,
         234,   222,  2813,   167,   227,   226, 23821,   118,   246,   167],
       device='cuda:0'), tensor([23433,  1320, 26397,   520,  2797], device='cuda:
0'), tensor([18308, 14179,  1222,   383, 13139,  7235,  9005,   532,   347,    12,
        26869,   352], device='cuda:0'), tensor([48230,    88, 17337,  3524,   532,
4403, 17084,  4198,  2232],
       device='cuda:0'), tensor([26699,  2550,  4551,  1105], device='cuda:0'), tens
or([   51, 33650,   532,   376,   549,   283], device='cuda:0'), tensor([   40,   36
7,  6158,  7013,   449, 45704,     0,  8959,   317, 13368,
        1677,   370,  1137,  6217,  3535,    37,     0], device='cuda:0'), tensor
([20917, 12911,   532, 15585, 19665,   357,    43,   346, 13329,    11,
        8377,   986], device='cuda:0'), tensor([  464,   520,   328, 31091,  3021,
25, 20787,   262, 49525],
       device='cuda:0'), tensor([10227, 44197,   362,   383, 22253,  3198,   410,  2
311,   308,    22],
       device='cuda:0'), tensor([37949,  2189, 33687,   319,   968, 30563,   329, 48
723,  4816],
       device='cuda:0'), tensor([   36, 43130,  8207,   338,  8362], device='cuda:
0'), tensor([38887, 42192,  7547,   319,  1605, 34392,    25,  9300,    73,   986],
       device='cuda:0'), tensor([   34,  9050, 12419, 11860,  1137,   327, 23518,  2
751,     0],
       device='cuda:0'), tensor([  464, 23762,  5155,  7922,   513,    25,   383, 23
762,  5155],
       device='cuda:0'), tensor([ 3792,   632,   317,  4599, 37560,  1675,  7631,
808,  1015, 36039,
        5985,   364,    30], device='cuda:0'), tensor([   50,  9229, 17582,   532,
366, 38318,   445, 30028,  1706,     1],
       device='cuda:0'), tensor([   33,   578,  2185,   532,   383, 19576,   338, 14
609, 21308,  7171,
        2644], device='cuda:0'), tensor([  464,  5675,  4216,  1526,   532,  2986,
9500, 28345],
       device='cuda:0'), tensor([29531, 25528,   286, 14927,  1108, 13879,   357,
940,    13,   940,
          13,  2998,     8], device='cuda:0'), tensor([34932,   236, 13783,   244,
20513, 26229, 12045,   233, 30640, 34460,
         106,   162,   235,   243], device='cuda:0'), tensor([   35, 11999,  6612,
532,  2490,  1299, 11853, 29505,  3717,  4551,
         362], device='cuda:0'), tensor([20916,   437,   367,  2518,     0, 17847,
11,  6889,   929,    11,
```

        30958,   11,   406, 1530, 1222,   517], device='cuda:0'), tensor([168,
226,   116,   167,   108,   242,   169,   222,   112,   412,
         1507,   657, 3829, 28362,   604], device='cuda:0'), tensor([29907, 15608,
6288, 8920,   317, 17746,   51, 1000, 7849, 7623,
        45231, 25043,   494, 2644], device='cuda:0'), tensor([31439, 20745,   362,
25, 1086,   480, 38782,   338, 3232,   12,
         1890,   12, 3237,   718,   266,   14, 9084,   320, 2127,   371],
       device='cuda:0'), tensor([72, 3629,   521,   82,   532,   968, 5270, 26
915, 28207],
       device='cuda:0'), tensor([9362, 2975, 38965,   352], device='cuda:0'), tens
or([1314,   400, 13641, 9570], device='cuda:0'), tensor([38, 2751, 4877, 841
0, 21515, 30065, 6561, 3228],
       device='cuda:0'), tensor([3666, 5155,   287, 40247,   11, 16123,   0], d
evice='cuda:0'), tensor([6109, 38573, 2764,   82,   25, 20580,   337, 14744, 38
93],
       device='cuda:0'), tensor([2484, 30188,   311,   654,   284, 10799, 42263], d
evice='cuda:0'), tensor([34, 3413, 1550, 40849,   25, 17084, 3864, 7465, 200
08, 29147,
        16656], device='cuda:0'), tensor([28566, 4307, 37810], device='cuda:0'), te
nsor([18102,   286, 8576,   337, 7211, 1039], device='cuda:0'), tensor([47,
967,   290, 49740], device='cuda:0'), tensor([31567,   375, 3681,   25, 4793, 149
62, 4913,   278, 4935],
       device='cuda:0'), tensor([45, 24112, 12509, 8579, 41333,   220, 32909, 14
732],
       device='cuda:0'), tensor([49717,   83, 3477,   89,   532,   347, 2007,
259, 37059,   357,
        11610,   72,   986], device='cuda:0'), tensor([49898,   338, 3943, 30958,
5438, 3050,   532, 25700,   309,   986],
       device='cuda:0'), tensor([41129,   403, 47142, 17801,   287, 6645], device
='cuda:0'), tensor([43224, 11182, 42192], device='cuda:0'), tensor([37590, 1872, 1
644, 12601, 15767, 25542], device='cuda:0'), tensor([49, 4449, 7697,   532, 413
20, 46678, 8949, 10117,   13, 26616],
       device='cuda:0'), tensor([2390, 1137, 25241,   6,   50, 41725,   309, 1
847, 3525,   532,
        48374, 14015, 41227, 6888, 6047], device='cuda:0'), tensor([3109, 5731,
25, 3964, 14484,   767, 34270, 34420],
       device='cuda:0'), tensor([41, 34907,   290,   509, 1286,   77,   338,
370, 3757,   448,
         7536], device='cuda:0'), tensor([15047, 15353,   554, 9363, 11470, 45949],
device='cuda:0'), tensor([15645, 4373,   532,   314, 1680, 26981, 26421, 2218,
986],
       device='cuda:0'), tensor([41191, 38573,   25, 21668, 3031,   284, 14728, 1
333, 2105,   986],
       device='cuda:0'), tensor([3666, 17252, 5249], device='cuda:0'), tensor([
38, 5681, 32060, 5675, 5356], device='cuda:0'), tensor([2437,   284,   787,   25
7, 14747,   276, 3290, 12187],
       device='cuda:0'), tensor([27082, 1797, 3268, 41722, 4146,   25,   383, 7
849, 7623],
       device='cuda:0'), tensor([21701, 11017, 13697, 26503, 14099, 9263, 36922, 13
298, 13700],
       device='cuda:0'), tensor([1925,   270, 30820,   25, 3497, 11459, 10216, 17
847,   0],
       device='cuda:0'), tensor([464, 12189, 33689, 1806, 25841], device='cuda:
0'), tensor([83, 14107,   499, 4875, 13004,   278,   532, 4096, 10375],
       device='cuda:0'), tensor([2437, 1675, 6889, 3232, 12911, 7467], device
='cuda:0'), tensor([25934,   362,   13,   17,   357,   37, 3287,   78,   8,

5313,
        34270, 8729,   986], device='cuda:0'), tensor([13856,   647, 10934,   82,
257, 10850], device='cuda:0'), tensor([9452,   346, 5119,   532, 25390,   357, 195
09, 2196,   8],
       device='cuda:0'), tensor([6719,   12, 6230, 21163,   379, 6016,   261], d
evice='cuda:0'), tensor([47, 3316,   12, 7975, 3228, 36923,   532, 9714,   3
11,   986],
       device='cuda:0'), tensor([14874, 31183,   262, 3801, 10572,   532,   366,
464, 12281,   986],
       device='cuda:0'), tensor([16,   13,   968, 6280, 4793, 15896,   287,
371,   671,   70,
          459, 3687,   11, 45238,   89,   11, 12873], device='cuda:0'), tensor
([32942, 5809, 1158, 1879, 3228], device='cuda:0'), tensor([2937, 7024,   25,
12737,   287, 2486, 1748,   11, 2869],
       device='cuda:0'), tensor([50,   55, 17887, 7623, 35911,   25, 3596, 4
930],
       device='cuda:0'), tensor([9704,   264, 20364, 46142, 1489,   519,   263,
319, 20052],
       device='cuda:0'), tensor([51, 4364, 10940,   89,   532, 13816,   317,
993, 10117,   13,
        14236,   349,   516,   685, 19238,   986], device='cuda:0'), tensor([21063,
3274, 3596,   286, 3961, 23102], device='cuda:0'), tensor([1925, 9140, 10301,
357, 16979,   12,   47, 2954, 2547, 1118,
           8], device='cuda:0'), tensor([29744,   376, 14992,   257, 7137,   532,
978, 1122, 4373, 1195,
           5,   32,   26], device='cuda:0'), tensor([54, 8845, 2439,   441,
8048,   718,   14, 1129,   14, 2931,
          513,   14, 1065,   357, 41275,   8], device='cuda:0'), tensor([35,
9728, 39221, 10204, 22724], device='cuda:0')], 'query': ['Talent Show Fail.', 'Kayak
ing in Der Wasser', 'Feeding America PSA with Matt Damon', 'Sleepover Awkwardness',
'Gulfstream II night landing view from cockpit jumpseat', 'How to Quit Smoking', 'Ko
be Bryant And Raja Bell Meeting For...', 'Graduating!', 'GTST classic 21 juni 2004',
'Chris Brown feat. Tyga - Holla At Me', 'Wanna Buy A Ghost?', 'Promma Mia 3/3', 'The
Ultimate Raw Fish', 'Jake and Amir: Girlfriend Pt. 2 (Aragorn)', 'Botchamania 77',
'The National - Live from Brooklyn on 5/15 (Trailer) - A V...', "World's 2nd Largest
Tetris Game - www.freshcreation.nl", '3d animation short "Bath Time"', 'Kites', 'Fea
rless customer tackles bank robber', 'THE BEAVER - Trailer', 'Fear Clinic - Entomoph
obia', 'Katrina Law Youtube Celebrity Playlist', 'something', 'Amazing Rube Goldberg
Fire Machine', 'Request: Makeup Starter Kit', "Assassin's Creed II Review", 'Awesome
Power Line Fire', 'Roger Federer Tweener Between The Leg...', '"TEENAGE DREAM" by Ka
ty Per...', "Tim McGraw's YouTube Playlist", 'Callejeros (cuatro) vecinas valencia',
'Jonathan Coulton - Baby Got Back...', "Ken's Dating Tips: Tip # 31", 'Mario: Game O
ver', 'Valeu - Exaltasamba', 'YouTube Symphony Orchestra Summit - Day 2!', 'American
Idol Star Adam Lambert At Gu...', 'AIR SUPPLY, sexiest shower scene ever...', 'Ramay
ana The Epic - The Legends', '100821 Adam Couple Kiss Cut', 'Madonna Sticky and Swee
t Sky1 commercial', 'Yummy Berry Cobbler - For Mom', 'Cambry plays Rachmaninoff Mome
nt Musical No. 4', "Brandon Jennings' NBA Journey", 'SUPERBOWL SUNDAY! (2/7/10-34
0)', 'TROPICAL SKITTLES (Full Version)', 'How to Wash A Cat - By Bud Herron', 'Emine
m - The Warning (Mariah Carey &...', 'The Joy of Gaming w/ SeaNanners: World of War
cr...', '"That\'s What Christmas Means To Me" Music Video To Benef...', '소녀시대 20
11년 캘', 'Try That Bike Stunt', 'Harry Potter & The Half Blood Prince - B-Roll
1', 'Buzzy Kerbox - Big Wave Surfer', 'Sadface Ep 12', 'Trenches - Fubar', 'I HATE Y
OU JOSH! IM A TEEN WEREWOLF!', 'Young Money - BedRock (Lil Wayne, Nic...', 'The Stig
Revealed: Behind the Scenes', 'HD Starcraft 2 TheLittleOne v Sen g7', 'Soccer Takes
on New Meaning for Haitian Team', "Ezekiel's Story", 'Lady Gaga Live on American Ido
l: Alej...', 'CITY SLICKER CAMPING!', 'The Beautiful Life Episode 3: The Beautiful L

ife', 'Is It A Good Idea To Microwave Pipe Cleaners?', 'Sonic Youth - "Sacred Trickster"', "Bite Me - The Gamer's Zombie Apocalypse Series...", 'The Grand Del Mar - San Diego Resort', 'Random Acts of Kindness Challenge (10.10.07)', '野外オナーニで逮捕', 'Darth Jackson - Britains Got Talent 2009 Ep 2', 'Weekend Haul! Skin, Makeup, Fashion, Lush & more', '세바퀴 E18 090808 4', 'Taylor Swift Today Was A FairyTale Music Video Ft Kellie...', "Modern Warfare 2: BlameTruth's Free-For-All 6 w/ Akimbo R", 'iBlinds - New generation iPod accessory', 'Off road Vid 1', '15th Century Email', 'GINGERS DO HAVE SOULS!!', 'My Life in Ruins, Really!', 'Every Minute Counts: Improve Maternal Health', 'Shaq Sings to Justin Bieber', 'Caught On Tape: Wave Crashes Into Cruise Ship', 'Internet Distractions', 'House of 1000 Muppets', 'Pork and Beans', 'Embodiment: 2006 Paper Journaling Project', 'Nissan LEAF™: Polar Bear', 'Dutchtasticz - Bitchin Trilogy (Origi...', "Victoria's Secret Fashion Show 2010 - Setting T...", 'Falun Gong Protest in NY', 'Asian Lady Gaga', 'Dubai police hunt Hamas killers', 'Rihanna - ROCKSTAR 101 ft. Slash', "AMERICA'S GOT TALENT -Kid singer Bianca Ryan", 'Exclusive: Windows Phone 7 Browser Comparison', "Jayson and Katelyn's Wipeout Date", 'Lightheaded In Store 360 Vinyl', 'Chris Brown - I Can Transform Ya feat...', 'Miami Minute: Dolphins respond to Jets tripping...', 'My pets combat', 'Greenscreen Grandmas', 'How to make a sculpted dog cake', 'PARIS IN JAIL: The Music Video', 'Camvas「サクラビト」', 'ChitChat: Get Clear Summer Skin!', 'The Self Stirring Mug', 'timbap digital DJing - basic interaction', 'How To Make Free Money Online', 'Android 2.2 (Froyo) Web Browser Speed...', 'Delmer Builds a Machine', 'Minilogue - Animals (short version)', 'Pre-season shred at brighton', 'Punch-Out!! Trailer - Nintendo S...', 'Schaffer the Darklord - "The Rap...', '1. New Year 2006 midnight in Radegast East, Lodz, Poland', 'Dog Drives Car!!', 'US elections: reactions in Obama city, Japan', 'SXSW Video Diary: Day Two', 'Mark sanchez wipes booger on teammate', 'Trey Songz - Say Aah ft. Fabolous [OF...', 'Quick First Day of School Hair', 'Chocolate Rain (Pop-Punk Parody)', 'Deep Frying a Turkey - Alton Brown Q&A;', 'WWE SmackDown 6/19/09 3/12 (HQ)', 'Dodge Challenger Freedom Commercial']}
****
****
****
****

1it [00:37, 37.84s/it]

{'input_ids': [tensor([31466,   298, 5438, 18448,   13], device='cuda:0'), tensor([37247,   868,   287, 9626,   370, 24929], device='cuda:0'), tensor([18332,   278, 2253,   350, 4090,   351, 4705, 33572],
       device='cuda:0'), tensor([40555, 2502, 5851, 12378, 1108], device='cuda:0'), tensor([38, 4754, 5532, 2873, 1755, 9581, 1570,   422, 25762, 4391,
        24073], device='cuda:0'), tensor([2437,   284, 48887, 39801], device='cuda:0'), tensor([42, 5910, 16754,   843, 13308,   64, 7459, 22244, 1114,   986],
       device='cuda:0'), tensor([42731, 11927,   0], device='cuda:0'), tensor([19555, 2257, 6833, 2310, 10891,   72, 5472], device='cuda:0'), tensor([15645, 4373, 2218,   13, 7039, 4908,   532, 11806,   64, 1629,
         2185], device='cuda:0'), tensor([54, 7697, 11763,   317, 9897,   30], device='cuda:0'), tensor([24129, 2611, 32189,   513,   14,   18], device='cuda:0'), tensor([464, 11165, 16089, 13388], device='cuda:0'), tensor([43930,   290, 43931,   25,   402, 9872, 34352,   13,   362,   357,
         3163,   363, 1211,   8], device='cuda:0'), tensor([20630, 49869, 5411, 8541], device='cuda:0'), tensor([464, 2351,   532, 7547,   422, 12232,   319,   642,   14, 1314,
          357, 15721, 5329,   8,   532,   317,   569,   986],
       device='cuda:0'), tensor([10603,   338,   362,   358,   406,   853,   395, 27351, 2442, 3776,
          532, 7324,   13, 48797, 38793,   13, 21283], device='cuda:0'), tensor([18,   67, 11034, 1790,   366,   33,   776, 3862,   1],
       device='cuda:0'), tensor([42, 2737], device='cuda:0'), tensor([37798, 1203, 6491, 18034, 3331, 29979], device='cuda:0'), tensor([10970, 9348,   32, 5959,   532, 36923], device='cuda:0'), tensor([37798, 26690,   532, 7232,   296, 19851], device='cuda:0'), tensor([25881, 22267, 3854, 27431, 46289, 3811, 4868], device='cuda:0'), tensor([18927], device='cuda:0'), tensor([42770,   371, 3266, 31225, 3764, 10850], device='cuda:0'), tensor([18453,   25, 6889,   929, 37460, 10897], device='cuda:0'), tensor([8021, 44961,   338, 24299, 2873, 6602], device='cuda:0'), tensor([49061, 4333, 6910, 3764], device='cuda:0'), tensor([43719, 10169, 11882, 24205,   877, 14307,   383, 3564,   986],
       device='cuda:0'), tensor([1, 9328, 1677, 11879,   360, 32235,   1,   416, 36715, 2448,
          986], device='cuda:0'), tensor([14967, 11130, 1831,   338, 7444, 3811, 4868], device='cuda:0'), tensor([34, 6765,   73, 27498,   357, 27399, 47756,   8, 1569, 17879,
          292, 1188, 29634], device='cuda:0'), tensor([30365, 27854, 1122,   532, 14801, 11853, 5157,   986],
       device='cuda:0'), tensor([27827,   338, 43528, 27558,   25, 23095, 1303, 3261],
       device='cuda:0'), tensor([42315,   25, 3776, 3827], device='cuda:0'), tensor([53, 1000,   84,   532, 1475, 2501,   292, 31842],
       device='cuda:0'), tensor([33869, 33195, 34017, 20014,   532, 3596,   362,   0],
       device='cuda:0'), tensor([7437, 34392, 2907, 7244, 36978, 1629, 1962,   986],
       device='cuda:0'), tensor([42149, 19549, 6489,   56,   11, 1714, 6386, 14643, 3715, 1683,
          986], device='cuda:0'), tensor([33754,   323, 2271,   383, 16781,   532,   383, 14270],
       device='cuda:0'), tensor([3064,   23, 2481, 7244, 15062, 1154, 20350, 9712],
       device='cuda:0'), tensor([18454, 6415,   520, 17479,   290, 15335, 5274,   16, 5068],
       device='cuda:0'), tensor([56, 13513, 20165, 31868, 1754,   532, 1114, 11254],

```
       device='cuda:0'), tensor([   34,  4131,   563,  5341,   371,   620,   805,
259,  2364, 29278,
        33375,  1400,    13,   604], device='cuda:0'), tensor([45467, 30593,     6,
7403, 15120], device='cuda:0'), tensor([40331,  1137,    33,  3913,    43, 35329, 26
442,     0,   357,    17,
          14,    22,    14,   940,    12, 23601,     8], device='cuda:0'), tensor([
5446,  3185, 20151, 14277,  2043, 14990,  1546,   357, 13295, 10628,
           8], device='cuda:0'), tensor([ 2437,   284, 22892,   317,  5181,   532,
2750, 10370,  2332,  1313],
       device='cuda:0'), tensor([   36,  1084,   368,   532,   383, 15932,   357,  1
526,  9520, 31612,
        1222,   986], device='cuda:0'), tensor([  464, 14087,   286, 14426,   266,
14,  6896,    45, 15672,    25,
        2159,   286,  1810,  6098,   986], device='cuda:0'), tensor([    1,  2504,
338,  1867,  6786, 28453,  1675,  2185,     1,   220,
        7849,  7623,  1675, 19899,   986], device='cuda:0'), tensor([  168,   228,
234,   167,   227,   222,   168,   233,   250,   167,
         234,   222,  2813,   167,   227,   226, 23821,   118,   246,   167],
       device='cuda:0'), tensor([23433,  1320, 26397,   520,  2797], device='cuda:
0'), tensor([18308, 14179,  1222,   383, 13139,  7235,  9005,   532,   347,    12,
        26869,   352], device='cuda:0'), tensor([48230,    88, 17337,  3524,   532,
4403, 17084,  4198,  2232],
       device='cuda:0'), tensor([26699,  2550,  4551,  1105], device='cuda:0'), tens
or([   51, 33650,   532,   376,   549,   283], device='cuda:0'), tensor([   40,   36
7,  6158,  7013,   449, 45704,     0,  8959,   317, 13368,
        1677,   370,  1137,  6217,  3535,    37,     0], device='cuda:0'), tensor
([20917, 12911,   532, 15585, 19665,   357,    43,   346, 13329,    11,
        8377,   986], device='cuda:0'), tensor([  464,   520,   328, 31091,  3021,
25, 20787,   262, 49525],
       device='cuda:0'), tensor([10227, 44197,   362,   383, 22253,  3198,   410,  2
311,   308,    22],
       device='cuda:0'), tensor([37949,  2189, 33687,   319,   968, 30563,   329, 48
723,  4816],
       device='cuda:0'), tensor([   36, 43130,  8207,   338,  8362], device='cuda:
0'), tensor([38887, 42192,  7547,   319,  1605, 34392,    25,  9300,    73,   986],
       device='cuda:0'), tensor([   34,  9050, 12419, 11860,  1137,   327, 23518,  2
751,     0],
       device='cuda:0'), tensor([  464, 23762,  5155,  7922,   513,    25,   383, 23
762,  5155],
       device='cuda:0'), tensor([ 3792,   632,   317,  4599, 37560,  1675,  7631,
808,  1015, 36039,
        5985,   364,    30], device='cuda:0'), tensor([   50,  9229, 17582,   532,
366, 38318,   445, 30028,  1706,     1],
       device='cuda:0'), tensor([   33,   578,  2185,   532,   383, 19576,   338, 14
609, 21308,  7171,
        2644], device='cuda:0'), tensor([  464,  5675,  4216,  1526,   532,  2986,
9500, 28345],
       device='cuda:0'), tensor([29531, 25528,   286, 14927,  1108, 13879,   357,
940,    13,   940,
          13,  2998,     8], device='cuda:0'), tensor([34932,   236, 13783,   244,
20513, 26229, 12045,   233, 30640, 34460,
         106,   162,   235,   243], device='cuda:0'), tensor([   35, 11999,  6612,
532,  2490,  1299, 11853, 29505,  3717,  4551,
         362], device='cuda:0'), tensor([20916,   437,   367,  2518,     0, 17847,
11,  6889,   929,    11,
       30958,    11,   406,  1530,  1222,   517], device='cuda:0'), tensor([  168,
```

```
226,   116,   167,   108,   242,   169,   222,   112,   412,
         1507,   657,  3829, 28362,   604], device='cuda:0'), tensor([29907, 15608,
6288,  8920,   317, 17746,    51,  1000,  7849,  7623,
        45231, 25043,   494,  2644], device='cuda:0'), tensor([31439, 20745,   362,
25,  1086,   480, 38782,   338,  3232,    12,
         1890,    12,  3237,   718,   266,    14,  9084,   320,  2127,   371],
       device='cuda:0'), tensor([   72,  3629,   521,    82,   532,   968,  5270, 26
915, 28207],
       device='cuda:0'), tensor([ 9362,  2975, 38965,   352], device='cuda:0'), tens
or([ 1314,   400, 13641,  9570], device='cuda:0'), tensor([   38,  2751,  4877,  841
0, 21515, 30065,  6561,  3228],
       device='cuda:0'), tensor([ 3666,  5155,   287, 40247,    11, 16123,     0], d
evice='cuda:0'), tensor([ 6109, 38573,  2764,    82,    25, 20580,   337, 14744,  38
93],
       device='cuda:0'), tensor([ 2484, 30188,   311,   654,   284, 10799, 42263], d
evice='cuda:0'), tensor([   34,  3413,  1550, 40849,    25, 17084,  3864,  7465, 200
08, 29147,
        16656], device='cuda:0'), tensor([28566,  4307, 37810], device='cuda:0'), te
nsor([18102,   286,  8576,   337,  7211,  1039], device='cuda:0'), tensor([   47,
967,   290, 49740], device='cuda:0'), tensor([31567,   375,  3681,    25,  4793, 149
62,  4913,   278,  4935],
       device='cuda:0'), tensor([   45, 24112, 12509,  8579, 41333,   220, 32909, 14
732],
       device='cuda:0'), tensor([49717,    83,  3477,    89,   532,   347,  2007,
259, 37059,   357,
        11610,    72,   986], device='cuda:0'), tensor([49898,   338,  3943, 30958,
5438,  3050,   532, 25700,   309,   986],
       device='cuda:0'), tensor([41129,   403, 47142, 17801,   287,  6645], device
='cuda:0'), tensor([43224, 11182, 42192], device='cuda:0'), tensor([37590,  1872,  1
644, 12601, 15767, 25542], device='cuda:0'), tensor([   49,  4449,  7697,   532, 413
20, 46678,  8949, 10117,    13, 26616],
       device='cuda:0'), tensor([ 2390,  1137, 25241,     6,    50, 41725,   309,  1
847,  3525,   532,
        48374, 14015, 41227,  6888,  6047], device='cuda:0'), tensor([ 3109,  5731,
25,  3964, 14484,   767, 34270, 34420],
       device='cuda:0'), tensor([   41, 34907,   290,   509,  1286,    77,   338,
370,  3757,   448,
         7536], device='cuda:0'), tensor([15047, 15353,   554,  9363, 11470, 45949],
device='cuda:0'), tensor([15645,  4373,   532,   314,  1680, 26981, 26421,  2218,
986],
       device='cuda:0'), tensor([41191, 38573,    25, 21668,  3031,   284, 14728,  1
333,  2105,   986],
       device='cuda:0'), tensor([ 3666, 17252,  5249], device='cuda:0'), tensor([
38,  5681, 32060,  5675,  5356], device='cuda:0'), tensor([ 2437,   284,   787,   25
7, 14747,   276,  3290, 12187],
       device='cuda:0'), tensor([27082,  1797,  3268, 41722,  4146,    25,   383,  7
849,  7623],
       device='cuda:0'), tensor([21701, 11017, 13697, 26503, 14099,  9263, 36922, 13
298, 13700],
       device='cuda:0'), tensor([ 1925,   270, 30820,    25,  3497, 11459, 10216, 17
847,     0],
       device='cuda:0'), tensor([  464, 12189, 33689,  1806, 25841], device='cuda:
0'), tensor([   83, 14107,   499,  4875, 13004,   278,   532,  4096, 10375],
       device='cuda:0'), tensor([ 2437,  1675,  6889,  3232, 12911,  7467], device
='cuda:0'), tensor([25934,   362,    13,    17,   357,    37,  3287,    78,     8,
5313,
```

34270, 8729,   986], device='cuda:0'), tensor([13856,   647, 10934,   82,
257, 10850], device='cuda:0'), tensor([9452,   346, 5119,   532, 25390,   357, 195
09, 2196,   8],
device='cuda:0'), tensor([6719,   12, 6230, 21163,   379, 6016,   261], d
evice='cuda:0'), tensor([47, 3316,   12, 7975, 3228, 36923,   532, 9714,   3
11,   986],
device='cuda:0'), tensor([14874, 31183,   262, 3801, 10572,   532,   366,
464, 12281,   986],
device='cuda:0'), tensor([16,   13,   968, 6280, 4793, 15896,   287,
371,   671,   70,
459, 3687,   11, 45238,   89,   11, 12873], device='cuda:0'), tensor
([32942, 5809, 1158, 1879, 3228], device='cuda:0'), tensor([2937, 7024,   25,
12737,   287, 2486, 1748,   11, 2869],
device='cuda:0'), tensor([50,   55, 17887, 7623, 35911,   25, 3596, 4
930],
device='cuda:0'), tensor([9704,   264, 20364, 46142, 1489,   519,   263,
319, 20052],
device='cuda:0'), tensor([51, 4364, 10940,   89,   532, 13816,   317,
993, 10117,   13,
14236,   349,   516,   685, 19238,   986], device='cuda:0'), tensor([21063,
3274, 3596,   286, 3961, 23102], device='cuda:0'), tensor([1925, 9140, 10301,
357, 16979,   12,   47, 2954, 2547, 1118,
8], device='cuda:0'), tensor([29744,   376, 14992,   257, 7137,   532,
978, 1122, 4373, 1195,
5,   32,   26], device='cuda:0'), tensor([54, 8845, 2439,   441,
8048,   718,   14, 1129,   14, 2931,
513,   14, 1065,   357, 41275,   8], device='cuda:0'), tensor([35,
9728, 39221, 10204, 22724], device='cuda:0')], 'query': ['Talent Show Fail.', 'Kayak
ing in Der Wasser', 'Feeding America PSA with Matt Damon', 'Sleepover Awkwardness',
'Gulfstream II night landing view from cockpit jumpseat', 'How to Quit Smoking', 'Ko
be Bryant And Raja Bell Meeting For...', 'Graduating!', 'GTST classic 21 juni 2004',
'Chris Brown feat. Tyga - Holla At Me', 'Wanna Buy A Ghost?', 'Promma Mia 3/3', 'The
Ultimate Raw Fish', 'Jake and Amir: Girlfriend Pt. 2 (Aragorn)', 'Botchamania 77',
'The National - Live from Brooklyn on 5/15 (Trailer) - A V...', "World's 2nd Largest
Tetris Game - www.freshcreation.nl", '3d animation short "Bath Time"', 'Kites', 'Fea
rless customer tackles bank robber', 'THE BEAVER - Trailer', 'Fear Clinic - Entomoph
obia', 'Katrina Law Youtube Celebrity Playlist', 'something', 'Amazing Rube Goldberg
Fire Machine', 'Request: Makeup Starter Kit', "Assassin's Creed II Review", 'Awesome
Power Line Fire', 'Roger Federer Tweener Between The Leg...', '"TEENAGE DREAM" by Ka
ty Per...', "Tim McGraw's YouTube Playlist", 'Callejeros (cuatro) vecinas valencia',
'Jonathan Coulton - Baby Got Back...', "Ken's Dating Tips: Tip # 31", 'Mario: Game O
ver', 'Valeu - Exaltasamba', 'YouTube Symphony Orchestra Summit - Day 2!', 'American
Idol Star Adam Lambert At Gu...', 'AIR SUPPLY, sexiest shower scene ever...', 'Ramay
ana The Epic - The Legends', '100821 Adam Couple Kiss Cut', 'Madonna Sticky and Swee
t Sky1 commercial', 'Yummy Berry Cobbler - For Mom', 'Cambry plays Rachmaninoff Mome
nt Musical No. 4', "Brandon Jennings' NBA Journey", 'SUPERBOWL SUNDAY! (2/7/10-34
0)', 'TROPICAL SKITTLES (Full Version)', 'How to Wash A Cat - By Bud Herron', 'Emine
m - The Warning (Mariah Carey &...', 'The Joy of Gaming w/ SeaNanners: World of War
cr...', '"That\'s What Christmas Means To Me" Music Video To Benef...', '소녀시대 20
11년 캘', 'Try That Bike Stunt', 'Harry Potter & The Half Blood Prince - B-Roll
1', 'Buzzy Kerbox - Big Wave Surfer', 'Sadface Ep 12', 'Trenches - Fubar', 'I HATE Y
OU JOSH! IM A TEEN WEREWOLF!', 'Young Money - BedRock (Lil Wayne, Nic...', 'The Stig
Revealed: Behind the Scenes', 'HD Starcraft 2 TheLittleOne v Sen g7', 'Soccer Takes
on New Meaning for Haitian Team', "Ezekiel's Story", 'Lady Gaga Live on American Ido
l: Alej...', 'CITY SLICKER CAMPING!', 'The Beautiful Life Episode 3: The Beautiful L
ife', 'Is It A Good Idea To Microwave Pipe Cleaners?', 'Sonic Youth - "Sacred Tricks

ter"', "Bite Me - The Gamer's Zombie Apocalypse Series...", 'The Grand Del Mar - San Diego Resort', 'Random Acts of Kindness Challenge (10.10.07)', '野外オナーニで逮捕', 'Darth Jackson - Britains Got Talent 2009 Ep 2', 'Weekend Haul! Skin, Makeup, Fashion, Lush & more', '세바퀴 E18 090808 4', 'Taylor Swift Today Was A FairyTale Music Video Ft Kellie...', "Modern Warfare 2: BlameTruth's Free-For-All 6 w/ Akimbo R", 'iBlinds - New generation iPod accessory', 'Off road Vid 1', '15th Century Email', 'GINGERS DO HAVE SOULS!!', 'My Life in Ruins, Really!', 'Every Minute Counts: Improve Maternal Health', 'Shaq Sings to Justin Bieber', 'Caught On Tape: Wave Crashes Into Cruise Ship', 'Internet Distractions', 'House of 1000 Muppets', 'Pork and Beans', 'Embodiment: 2006 Paper Journaling Project', 'Nissan LEAF™: Polar Bear', 'Dutchtasticz - Bitchin Trilogy (Origi...', "Victoria's Secret Fashion Show 2010 - Setting T...", 'Falun Gong Protest in NY', 'Asian Lady Gaga', 'Dubai police hunt Hamas killers', 'Rihanna - ROCKSTAR 101 ft. Slash', "AMERICA'S GOT TALENT -Kid singer Bianca Ryan", 'Exclusive: Windows Phone 7 Browser Comparison', "Jayson and Katelyn's Wipeout Date", 'Lightheaded In Store 360 Vinyl', 'Chris Brown - I Can Transform Ya feat...', 'Miami Minute: Dolphins respond to Jets tripping...', 'My pets combat', 'Greenscreen Grandmas', 'How to make a sculpted dog cake', 'PARIS IN JAIL: The Music Video', 'Camvas「サクラビト」', 'ChitChat: Get Clear Summer Skin!', 'The Self Stirring Mug', 'timbap digital DJing - basic interaction', 'How To Make Free Money Online', 'Android 2.2 (Froyo) Web Browser Speed...', 'Delmer Builds a Machine', 'Minilogue - Animals (short version)', 'Pre-season shred at brighton', 'Punch-Out!! Trailer - Nintendo S...', 'Schaffer the Darklord - "The Rap...', '1. New Year 2006 midnight in Radegast East, Lodz, Poland', 'Dog Drives Car!!', 'US elections: reactions in Obama city, Japan', 'SXSW Video Diary: Day Two', 'Mark sanchez wipes booger on teammate', 'Trey Songz - Say Aah ft. Fabolous [OF...', 'Quick First Day of School Hair', 'Chocolate Rain (Pop-Punk Parody)', 'Deep Frying a Turkey - Alton Brown Q&A;', 'WWE SmackDown 6/19/09 3/12 (HQ)', 'Dodge Challenger Freedom Commercial'], 'response': [' Despite this, it', ' River at Großynden', ' during halftime of Patriots-', ' Gotan together with', ' feet from garbage can.\n\n', ' for Better Health\n\n1 Bill', ' Free View in iTunes\n\n130', ' The hoodie that fits me reads', '\n\nRoute Description:', ' (feat. Young Thug)', ' See Full List\n\n', '/2016 4. UPDATE:', ' Shell\n\nI', ' [Video Game] [Erot', ' Sunbathe Permaf', ' Free View in iTunes', '\n\nF1', ' in which a s', " are Here [D 'Lake", '\n\nEmployee video', ' 1\n\n| |\n', ' 144 4 Dartmouth College Special Education 2', '\n\nStep Rockbell - Lover', '.com/story/news', '" is a storm', '\n\nFree Expedited', ' 26+,720p YouTube', ' Alarm Rifles\n', ' 1017 Micro Camera Dar', ' Free View in iTunes', '\n\nHTML for Play', ' colocar 9', '\n\nArt and cover by BRE', ' – Complete List Kiss', ' (Diddy Kong Racing)\n', ' ------------------------------+ --- ----------', '\n\nBass Coast:', '\n\nSunday, August 04,', ' [groans] Mmmm,', ' and Legends, Vol', ': 5/28/2017', ').\n\nWhat do you think', ' - The Mentales', ' (London/Rosemond', ': Year 1 (Quint-', '\n\nAfter the festivities', '\n\n\nQUICK LINK: Inv', " Real cats don't need baths.", ' Free View in iTunes\n\n', ' Free View in iTunes\n\n19', ' See more Hot 47', '\'\'소', ' at Metrorail while', 'st January 2003\n\n', ' (99) info@catalog', '57 The Frantics', 'Trench -Alongland Dark Forest', ' DELIRIUM SETS ME', ' 2 Miraclemen 2', ' Of The Star Spangled Banner\n', ' v MachiMt6 (', '\n\nSounders FC Stub', '" is "the essence of the', ' USA (USA) November 23', 'ING!\n\n!]]<|endoftext|>', ' Episode 3: The', ' In this article we save all', '\n\n66.', ' Apocalypse Series...<|endoftext|>', ' and Casino\n\n', '\n\n\nWell, here', 'する。\'', ' (https://youtu.be', '! https://t', '59 04/11', ' Was Actually 960 Degrees', 'ancor @ Karachi Sumof-', ' set with 16gb', ' (below) plays,', ' Scam (Blog)\n', ' ** Phoenix: * So', ' The Mama Puff Hat, the', ' Through BabyCorps', "'s Justin Bieber, Re", ' in Japan\n\nIn', '. Or maybe you are an entrepreneur', '" parody stings?', ' with Wild Rice. 6', '. Duplication authorized with approval from', '™,; iConnectZ', '\n\nNew Wave - Transatlantic', " Victoria's Secret Fashion Show 2011 -", '\n\n4-Oct-', '.\n\nWeathered billionaire', "\n\n'The second time we", '! [Domino', ' gave a 29-minute set', "\n\nWe've rounded up the", ': Oct 5th,', ' Records\n\nStart', ' TLC Wilson Phillips -', "Dolphins' Ryan Tanne", ' value increased by 4', " even think

s vampires won't stop their", ' Jamie Tapp 5', ' for Courtney Barnett\'s "S', 'のよう', '\n\nRELEASEDwow4', ' has been re-printed under licence', ' interaction design.<|endoftext|>', '\n\nThere are so many ways', 'Apr 20, 2018', ' (by E.', 'short version)<|endoftext|>', ' zu Schaff', ' (2001)\n\nCritical', ' Free View in iTunes\n\n84', ' First time since 9', '... Kreygasm - Bro', '\n\nSource: CNN.', "\n\nSXSW 2014's", "'s face during The Sound of Music", ' Free View in iTunes\n', 'cut\n\nBreak in', '\n\n7. Blu', ' Leicester/Alter', ' Please select yes. Pre', ' with a Drag Strip Test']}
1
{'input_ids': [tensor([2061, 284, 2822, 262, 1466, 287, 534, 1204, 532, 2644],
       device='cuda:0'), tensor([3041,   25, 2488, 47095, 1326,   499, 1647, 930, 5706, 43537],
       device='cuda:0'), tensor([43410, 16706,   406, 22436, 36557, 6416, 14213, 357, 13807, 1303,
          16,   8], device='cuda:0'), tensor([31443,   88, 12579], device='cuda:0'), tensor([45708,   805,   532, 9935,   338, 2892, 39795, 1475, 17040,   25,
         986], device='cuda:0'), tensor([2396, 314, 1138, 337, 9618, 329, 1103,   89,   0], device='cuda:0'), tensor([18858, 20492, 2199, 47714,   532,   366,   464, 37123,   437,   986],
       device='cuda:0'), tensor([43471, 2414,   25,   383, 27330, 4631, 10243], device='cuda:0'), tensor([9915, 26123,   25, 3232,   329, 1439,   352, 1222,   362],
       device='cuda:0'), tensor([41, 29309, 19013, 3691, 4380,   319,   262, 8511],
       device='cuda:0'), tensor([46678, 43236,   42,   532, 31900, 7623, 3776, 44031],
       device='cuda:0'), tensor([5195,   466,   345, 2342,   262, 5861,   314, 17266,   986],
       device='cuda:0'), tensor([1858, 2561, 1355, 10370], device='cuda:0'), tensor([6398,   84, 32009, 18840, 7088, 4712,   509, 13299,   11, 8525,
         6184,   223,   70, 5799,   513,   13, 1065,   13, 10333],
       device='cuda:0'), tensor([464, 15640,   286, 9827,   290, 3851,   986], device='cuda:0'), tensor([43879, 4525,   317, 39108], device='cuda:0'), tensor([464,   327,   330, 10546,   532, 31899,   470, 11853, 13111,   286,
         2644], device='cuda:0'), tensor([11531,   324, 2879, 20201,   968, 1971, 2947, 12058],
       device='cuda:0'), tensor([5568, 7197], device='cuda:0'), tensor([2043,   45,   25, 2869, 11881,   705, 6888, 2664,   645, 2328,
          6], device='cuda:0'), tensor([464, 367, 1942, 1452, 278], device='cuda:0'), tensor([45565,   0, 45565,   0], device='cuda:0'), tensor([47, 21358, 9878, 8924, 42743], device='cuda:0'), tensor([12050,   257, 3580], device='cuda:0'), tensor([818,   262, 1097,   351, 3700, 1737,   532, 5849, 10740,   532,
         7823], device='cuda:0'), tensor([18122, 2561, 1355, 14446,   532, 5325, 377,   709,   14, 3163,
          77, 1252], device='cuda:0'), tensor([8017,   680, 4287, 6374, 4339, 532, 10031, 2848],
       device='cuda:0'), tensor([3955, 25922,   287,   257, 17012], device='cuda:0'), tensor([45, 276, 352, 5601, 657], device='cuda:0'), tensor([47526,   51, 39484, 5097, 1546, 1222, 11179, 10116,   56, 29377,
        12599, 2751], device='cuda:0'), tensor([38, 3014, 2097, 12648, 1222, 9440, 10979, 22904, 6465,   220,
         5525,   250,   246,   164,   249,   249], device='cuda:0'), tensor([320, 324, 2788, 1042, 396, 539, 532, 7933, 1079, 328, 5355, 2644],
       device='cuda:0'), tensor([37,   615,   260,   520, 30915, 2750, 9957, 287,   257, 10299,
         259, 13590,   12, 28628, 2644], device='cuda:0'), tensor([36534, 28683,

```
10299,  3383], device='cuda:0'), tensor([ 1157, 44923,   284, 19175,   329,  4956,
287,   399,   986],
       device='cuda:0'), tensor([16635,   623,  3320, 36476,  1455,    13,   785, 35
651,    25,   520,
         676, 32604,   263, 23896,   338,  4021,   372,   338,   366,    45],
       device='cuda:0'), tensor([    1, 10943, 48842,     1,  3310,   524,  9794,
11, 13496,   385,
        2159,  8284,    11,  9910,    64,  9910,    64,    11, 13596,   948],
       device='cuda:0'), tensor([46602,  3764,  7738,   532,  2142,  1596], device
='cuda:0'), tensor([21466, 26544,  4434,    64,  9232, 23997,  1535, 27558,   351,
607,
        2644], device='cuda:0'), tensor([25881,    88, 14105,  6130,   366,  6435,
330,   735,     1],
       device='cuda:0'), tensor([ 4625,  7050, 22948,   865,   265,  3044,  4170], d
evice='cuda:0'), tensor([14214,    69,  3083,    25,   383, 23181,  6407, 28032,   2
86,   262,
         350,  2093, 20951,  7884], device='cuda:0'), tensor([   47,  1404,  2885,
1340,  5550, 10560,  3525, 13909, 12964, 17852,
         350, 25994,    46,   317,  4725,   449,   986], device='cuda:0'), tensor
([4666,  276], device='cuda:0'), tensor([ 6767,   641,  6889, 26855,   309,    12,
2484,  2265,   532, 10358,
         632,  1355,   347,  3577,    30], device='cuda:0'), tensor([   39,  2217,
7897, 29061,   364, 10185,   357,    23,    13,  3132,
          13,  2931,   532,   360,   986], device='cuda:0'), tensor([ 9148,  3913,
449,  9864, 39947,    57,  4146, 16868,  3563,    56,
        2538,    30], device='cuda:0'), tensor([43930,   338, 23102,  8968], device
='cuda:0'), tensor([31439, 20745,   362,  1058, 34198, 25737,  7922,   352,   532, 1
1140,
        1222], device='cuda:0'), tensor([22125,  7336, 44423,  3336,   376,  3843,
11335,   532, 34958,   406,
       49494,   569,   986], device='cuda:0'), tensor([   50,  8845,  2767,   309,
28082], device='cuda:0'), tensor([50241, 31376,    25,  3497,  3806,   383, 19175],
device='cuda:0'), tensor([ 3123,  1052,   377,   272,   555, 26849,   257, 24568, 10
115, 36309,
         390, 15095, 10094,   986], device='cuda:0'), tensor([ 2061,   262, 25003,
12248,    30,   351,   509,  6981,  6500],
       device='cuda:0'), tensor([39152,   390,  4881, 17364,   572,   355, 21166, 20
070,   329, 21933],
       device='cuda:0'), tensor([16157,   604,   400, 16134, 10185], device='cuda:
0'), tensor([16775,    25, 19100,  3050,    25, 17905,   422,   262, 42754],
       device='cuda:0'), tensor([47379, 39941,   351,  2561, 12880, 11252], device
='cuda:0'), tensor([    1,  2504,   338, 16013,     1,  4849,  1769, 16013, 22520,
12,
       15307, 42495], device='cuda:0'), tensor([14254,  5896], device='cuda:0'), te
nsor([32470, 19705,  1004,    65,  1313, 17906,   354,   317, 17001],
       device='cuda:0'), tensor([ 5962, 18720,    19,   983,   287,   257,   981,
357, 19419, 14604,
           8], device='cuda:0'), tensor([  72, 3987,  470], device='cuda:0'), tenso
r([49898,  4502,   532,   366,  8048,  2080,  5896,     1],
       device='cuda:0'), tensor([21868,  3691,   360,   967,   805], device='cuda:
0'), tensor([20238,  1374,  1675,  6869, 11152], device='cuda:0'), tensor([   35,  3
861,  7336, 44253,  6465,    56,  2538,  6226,  6684,    37,
        3228], device='cuda:0'), tensor([   1, 2061, 1002,    1,  532, 6416, 1204,
550, 2008,  983, 2128, 3048],
       device='cuda:0'), tensor([41173,     6,    51,   309, 15675], device='cuda:
0'), tensor([41762,   364,   911,  2096,  1550,   383,   350,   808,    75],
```

device='cuda:0'), tensor([38, 7197, 532, 7610, 1550, 7849, 7623], device
='cuda:0'), tensor([48412,   53, 8270, 16036,   532, 2365, 1315,   400],
device='cuda:0'), tensor([34, 3861,   57,   56,   347, 31949, 1546, 3
228],
device='cuda:0'), tensor([31660,   163,   110,   240, 36365,   112,   163,
237,   254,   162,
236,   231, 34460,   110,   165,   238,   113,   162,   110,   247],
device='cuda:0'), tensor([45708,   805,   532, 7542, 5476,   361,   993,
319, 3899, 6612],
device='cuda:0'), tensor([3351,   631, 10813,   775, 48038,   532, 5896], d
evice='cuda:0'), tensor([35, 20474, 44322, 9212, 16013, 5948, 1987,   400, 29
32, 7769,
532, 2644], device='cuda:0'), tensor([18081,   88, 13840,   329, 2444,
290,   584, 2644],
device='cuda:0'), tensor([24151, 8953,   656, 29420,   981, 36634], device
='cuda:0'), tensor([8264, 22707,   338,   371,   13, 5741, 5256, 1326,   88,
11,
12655,   319, 8378,   286, 1675,   986], device='cuda:0'), tensor([15017,
31045, 3764,   860,   12,   24,   12,   940],
device='cuda:0'), tensor([46751, 8407,   532, 12169, 39348], device='cuda:
0'), tensor([39, 2586,   329, 18509, 2586], device='cuda:0'), tensor([12442, 39
139, 9252,   532, 7922, 1248], device='cuda:0'), tensor([13256, 6612,   532,   3
83, 5896,   921, 12793], device='cuda:0'), tensor([51, 8254, 16483, 27433, 509
4, 21983], device='cuda:0'), tensor([2662,   38, 5181, 16860, 34017, 18448], devi
ce='cuda:0'), tensor([42,   437, 5557, 29618, 6056, 5250, 15626,   319, 7631,
986],
device='cuda:0'), tensor([7376, 9259, 12569, 41527, 31772,   0], device
='cuda:0'), tensor([5962, 7755, 10682], device='cuda:0'), tensor([13448, 9659, 7
967, 27794, 5075], device='cuda:0'), tensor([2257, 2200, 2767, 39484, 2149,   5
13], device='cuda:0'), tensor([49757, 38923, 23624, 9986], device='cuda:0'), tensor
([48640, 1222, 44485, 15253], device='cuda:0'), tensor([5211,   407, 9870, 13118,
20051,   0, 3409, 19155,   986],
device='cuda:0'), tensor([9915, 21566,   276, 2631,   292,   532,   314,
402, 12375, 39635,
357, 10913,   986], device='cuda:0'), tensor([18102,   286, 8576,   337,
7211, 1039], device='cuda:0'), tensor([22973,   382,   532, 4280, 1098,   685, 27
977, 2149, 12576, 35507,
60], device='cuda:0'), tensor([7206, 2885, 24027,   262, 1642,   286],
device='cuda:0'), tensor([3546, 2611, 3497, 632, 554, 532, 2520, 8511, 5521, 44
8,   0],
device='cuda:0'), tensor([29907, 22233,   11, 1248, 2709, 1756, 8533,
532, 2253,   338,
11853, 17388,   986], device='cuda:0'), tensor([10970, 9440, 6489, 3955,
15365, 7375, 3069, 6242],
device='cuda:0'), tensor([27722,   12,   51, 1726,   262, 3000, 1303, 1
485,   25, 5059,
13, 37727,   13, 25635,   13,   357,   701,   13,   775, 8471],
device='cuda:0'), tensor([46827, 8314, 4849,   89, 7423], device='cuda:
0'), tensor([38,   859, 3876, 19147], device='cuda:0'), tensor([49174,   616, 1
097], device='cuda:0'), tensor([23812, 3406, 4942,   68,   25, 3738, 42722, 159
90, 1559, 1220,
15585, 2644], device='cuda:0'), tensor([43, 9632,   406, 1590, 1802,
23, 18360, 5056,   265,   11,
290,   350, 48809], device='cuda:0'), tensor([33, 31777,   338, 8407,
33233,   532,   220,   943, 7012, 1031,
11, 31995,   257,   986], device='cuda:0'), tensor([14157, 5913], device

='cuda:0'), tensor([3855, 3406, 7623,   319, 7092, 23656,   0], device='cuda:0'), tensor([2061,   262, 25003, 12248,   30,   351,   509, 6981, 6500],
       device='cuda:0'), tensor([4535, 7340, 6535,   56, 40342,   53, 25633,   402, 4663, 6561,
            0,   357, 1157,   14, 2078,   14, 2931,   12, 26276,   8],
       device='cuda:0'), tensor([50,   624,   286,   921,   532, 7257, 7336,   357, 28529, 7623,
            8], device='cuda:0'), tensor([45, 8874, 18694, 8874,   220, 3806, 2975, 7930,   400, 27752,
        12873, 3717], device='cuda:0'), tensor([34,   372, 2645, 11768,   532, 34920,   770,   930, 6462, 8829,
        29496,   357, 41275,   8], device='cuda:0'), tensor([3791, 32052,   350,   873,   286, 6997, 3442,   290,   49,   72,
          986], device='cuda:0'), tensor([49, 4449, 7697,   25, 2011, 33869], device='cuda:0'), tensor([13965, 40195, 49224, 5565, 42399, 31091, 3021,   532, 42700, 2813,
         2644], device='cuda:0'), tensor([33869,   338, 18881,   805,   338, 50125,   341], device='cuda:0'), tensor([464, 6785, 1114, 4687], device='cuda:0'), tensor([34720, 9936, 6006, 10067, 3180,   311, 6998, 24218, 8035, 13896],
       device='cuda:0'), tensor([25433, 50133,   539, 9280], device='cuda:0'), tensor([22797, 1869,   362], device='cuda:0'), tensor([464, 350, 414, 5172], device='cuda:0'), tensor([21077,   25, 9164, 20051,   286,   968, 1971, 20206],
       device='cuda:0'), tensor([7414,   789, 20745,   532, 18773, 24015,   291, 7561,   311,   986],
       device='cuda:0'), tensor([3109, 9124,   653, 5478,   25, 21167,   262, 5905, 28089,   532,
        28697], device='cuda:0')], 'query': ['What to buy the women in your life -...', 'Re: @knitmeapony | Old Spice', 'Lindsay Lohan Needs Real Friends (Ep #1)', 'Rainy Days', "Letterman - Dave's Monologue Excerpt:...", 'So I met Miley for realz!', 'Ann Marie Calhoun - "The Pretend...', 'Mega64: The Beatles Rock Band', 'Black Ops: Free for All 1 & 2', 'Jumbo Jet vs People on the Beach', 'STAR TREK - Angry Video Game Nerd', 'Why do you watch the videos I ma...', 'There Will Be Bud', 'Actuación Insula Kampa, Pub Ágora 3.12.2010', 'The Adventures of Batman and Rob...', 'Float Like A Butterfly', "The Cacaman - Ain't Got Much of...", 'Paladino threatens New York Post Editor', 'kaylee', "ITN: Japan radiation 'causes no concern'", 'The Hauntening', 'Chicken!Chicken!', 'PALE BLUE DOT', 'Make a difference', 'In the car with James May - Top Gear - BBC', 'History Will Be Made - Radulov/Arnott', 'Polish Police SCBA - Chopper', 'IMAX in a basement', 'Ned 1 Den 0', 'BUTT MUSCLES & HEAVY BREATHING', 'Giant House Spider & COMMENT CONTEST 蜘蛛', 'imadethismistake - gravediggers...', 'Favre Struck By Football in a Groin Shot- Says...', 'Baby Bunny Grooming', '11 Reasons to Vote for Democrats in N...', 'BoondocksBootleg.com Exclusive: Stinkmeaner Remix\'s Usher\'s "N', '"CONNECT" Regime Change, Negus World Order, Saba Saba, Narcy', 'Pokémon FireRed - Part 17', 'Style Icon Malaika Shares health Tips with her...', 'Katy Perry talks "Peacock"', 'underwater rugby bratislava', 'Surfwise: The Amazing True Odyssey of the Paskowitz Family', 'PATADON DE DRENTHE EN EL PECHO A UN J...', 'moded', 'Teens Make Offensive T-Shirt - Should It Be Banned?', 'Huge Spanish Knockers!!! (8.31.09 - D...', 'BLOW JOB BRAZILIAN STYLE?', "Jake's Haircut", 'Modern Warfare 2 : Shotgun Sally Episode 1 - Search &', 'NIKE WRITE THE FUTURE - FULL LENGTH V...', 'SWEET TALK', 'Terry Tate: Get Out The Vote', 'Le Anulan un Gol a Cristiano Ronaldo de globito...', 'What the WHAT?!? with KANDEE', 'Tour de France kicks off as Armstrong pushes for comeback', 'July 4th parade!!!', 'Project:Report 2010: Reports from the Winners', 'BAT FIGHT with Will Ferrell', '"That\'s Gay" Salutes Gay Reality-Show Judges', 'Twitter Love', 'NBA Finals Lebron Etch A Sketch', 'First Cod4 game in a while (without commentary)', "iDon't", 'Victoria George - "Down With Love"', 'Ryan vs Dorkman', 'Learn How To Moonwalk', 'DRAKE FREESTYLE SPOOF!!', '"What If" - Real life had video game sound effects', "DON'T TREAD", 'Transformers Shorts On The Pro

wl', 'Glee - Dream On Music Video', 'FarmVille Podcast - Jan 15th', 'CRAZY BITCHE
S!!', '一粒水珠掉進鐵沙', 'Letterman - Queen Latifah on Michael Jackson', 'Screeching
Weasel - Love', 'DMG Dill Mill Gayye 24th August 09 -...', 'Healthy meals for studen
ts and other...', 'Girl falls into fountain while texting', "GEICO's R. Lee Ermey, a
ppearing on behalf of To...", 'San Bruno Fire 9-9-10', 'Jacob Golden - Zero Integrit
y', 'Homes for Gnomes', 'Super Luigi Galaxy - Episode 18', 'Michael Jackson - The Lo
ve You Save', 'Tiger Woods Makes Public Statement', 'OMG Cat watches Orchestra Fai
l', 'Kendrick Perkins flagrant foul on Mic...', 'Cheesy Ringtone Songs!', 'First Per
son Mario', 'Matt Allen Snowboarding 2005', 'STREET MUSIC 3', 'Moscow Subway Traffic
Jam', 'Emily & Henrietta', 'Do not Trust Profile Pictures! Samsun...', 'Black Eyed P
eas - I Gotta Feeling (FR...', 'House of 1000 Muppets', 'Paramore - Decode [OFFICIAL
VIDEO]', 'DEADLINE the making of', 'Imma Get It In - South Beach Workout!', "Taylor
Matthews, 18 auditions Dallas - America's Got Tale...", 'THE COMPLIMENTS COLLAB', 'A
uto-Tune the News #13: driving. stripping. swinging. (ft. Weez', 'Maria Does Salzbur
g', 'Grammar Nazis', 'Destroy my car', 'Know Your Meme: Antoine Dodson / Bed...', 'L
ogan Lacy 1008 lb Squat, and Puked', "Bollywood's Golden Triangle - Arbaaz, Salman
a...", 'North Carolina', 'Get Your Video on SportsCenter!', 'What the WHAT?!? with K
ANDEE', 'NAUGHTY ELEVATOR GIRLS! (11/28/09-269)', 'Sick of You - CAKE (Official Vide
o)', 'NOVIKOV Out road 66th Rally Poland 2009', 'Cheryl Cole - Promise This | Full
Radio Rip (HQ)', 'New Dirty Pics of Ms California andRi...', 'Rihanna: MyYouTube',
'NEW Toshiba Android Tablet Revealed - CES 2011...', "YouTube's Layman's Explanatio
n", 'The Quest For Space', 'DIRTY SHAME IS SRS BSNS!!!!', 'milkshake dance', 'Iron M
an 2', 'The Pity Card', 'Found: Lost Pictures of New York Blizzard', 'Flower Warfare
- Psychedelic Action S...', 'Expedition Africa: Meet the Explorers - Benedict']}
****
****
****
****

1it [00:53, 53.90s/it]

{'input_ids': [tensor([2061, 284, 2822, 262, 1466, 287, 534, 1204, 532, 2644],
       device='cuda:0'), tensor([3041,   25, 2488, 47095, 1326,   499, 1647,
930, 5706, 43537],
       device='cuda:0'), tensor([43410, 16706,   406, 22436, 36557, 6416, 14213,
357, 13807, 1303,
          16,   8], device='cuda:0'), tensor([31443,   88, 12579], device='cuda:
0'), tensor([45708,   805,   532, 9935,   338, 2892, 39795, 1475, 17040,   25,
         986], device='cuda:0'), tensor([2396, 314, 1138, 337, 9618, 329, 1103,
89,   0], device='cuda:0'), tensor([18858, 20492, 2199, 47714,   532,   366,   46
4, 37123,   437,   986],
       device='cuda:0'), tensor([43471, 2414,   25,   383, 27330, 4631, 10243], d
evice='cuda:0'), tensor([9915, 26123,   25, 3232,   329, 1439,   352, 1222,   3
62],
       device='cuda:0'), tensor([41, 29309, 19013, 3691, 4380,   319,   262, 8
511],
       device='cuda:0'), tensor([46678, 43236,   42,   532, 31900, 7623, 3776, 44
031],
       device='cuda:0'), tensor([5195,   466,   345, 2342,   262, 5861,   314, 17
266,   986],
       device='cuda:0'), tensor([1858, 2561, 1355, 10370], device='cuda:0'), tens
or([6398,   84, 32009, 18840, 7088, 4712,   509, 13299,   11, 8525,
        6184,   223,   70, 5799,   513,   13, 1065,   13, 10333],
       device='cuda:0'), tensor([464, 15640,   286, 9827,   290, 3851,   986], d
evice='cuda:0'), tensor([43879, 4525,   317, 39108], device='cuda:0'), tensor([46
4,   327,   330, 10546,   532, 31899,   470, 11853, 13111,   286,
        2644], device='cuda:0'), tensor([11531,   324, 2879, 20201,   968, 1971,
2947, 12058],
       device='cuda:0'), tensor([5568, 7197], device='cuda:0'), tensor([2043,   4
5,   25, 2869, 11881,   705, 6888, 2664,   645, 2328,
         6], device='cuda:0'), tensor([464, 367, 1942, 1452, 278], device='cud
a:0'), tensor([45565,   0, 45565,   0], device='cuda:0'), tensor([47, 21358,
9878, 8924, 42743], device='cuda:0'), tensor([12050,   257, 3580], device='cuda:
0'), tensor([818,   262, 1097,   351, 3700, 1737,   532, 5849, 10740,   532,
        7823], device='cuda:0'), tensor([18122, 2561, 1355, 14446,   532, 5325,
377,   709,   14, 3163,
          77, 1252], device='cuda:0'), tensor([8017,   680, 4287, 6374, 4339,
532, 10031, 2848],
       device='cuda:0'), tensor([3955, 25922,   287,   257, 17012], device='cuda:
0'), tensor([45, 276, 352, 5601, 657], device='cuda:0'), tensor([47526,   51,
39484, 5097, 1546, 1222, 11179, 10116,   56, 29377,
       12599, 2751], device='cuda:0'), tensor([38, 3014, 2097, 12648, 1222,
9440, 10979, 22904, 6465,   220,
        5525,   250,   246,   164,   249,   249], device='cuda:0'), tensor([320,
324, 2788, 1042, 396, 539, 532, 7933, 1079, 328, 5355, 2644],
       device='cuda:0'), tensor([37,   615,   260,   520, 30915, 2750, 9957,
287,   257, 10299,
         259, 13590,   12, 28628, 2644], device='cuda:0'), tensor([36534, 28683,
10299, 3383], device='cuda:0'), tensor([1157, 44923,   284, 19175,   329, 4956,
287,   399,   986],
       device='cuda:0'), tensor([16635,   623, 3320, 36476, 1455,   13,   785, 35
651,   25,   520,
         676, 32604,   263, 23896,   338, 4021,   372,   338,   366,   45],
       device='cuda:0'), tensor([1, 10943, 48842,   1, 3310,   524, 9794,
11, 13496,   385,
        2159, 8284,   11, 9910,   64, 9910,   64,   11, 13596,   948],
       device='cuda:0'), tensor([46602, 3764, 7738,   532, 2142, 1596], device

```
='cuda:0'), tensor([21466, 26544,  4434,    64,  9232, 23997,  1535, 27558,   351,
607,
         2644], device='cuda:0'), tensor([25881,    88, 14105,  6130,   366,  6435,
330,   735,     1],
       device='cuda:0'), tensor([ 4625,  7050, 22948,   865,   265,  3044,  4170], d
evice='cuda:0'), tensor([14214,    69,  3083,    25,   383, 23181,  6407, 28032,   2
86,   262,
          350,  2093, 20951,  7884], device='cuda:0'), tensor([   47,  1404,  2885,
1340,  5550, 10560,  3525, 13909, 12964, 17852,
          350, 25994,    46,   317,  4725,   449,   986], device='cuda:0'), tensor
([4666,  276], device='cuda:0'), tensor([ 6767,   641,  6889, 26855,   309,    12,
2484,  2265,   532, 10358,
          632,  1355,   347,  3577,    30], device='cuda:0'), tensor([   39,  2217,
7897, 29061,   364, 10185,   357,    23,    13,  3132,
           13,  2931,   532,   360,   986], device='cuda:0'), tensor([ 9148,  3913,
449,  9864, 39947,    57,  4146, 16868,  3563,    56,
         2538,    30], device='cuda:0'), tensor([43930,   338, 23102,  8968], device
='cuda:0'), tensor([31439, 20745,   362,  1058, 34198, 25737,  7922,   352,   532, 1
1140,
         1222], device='cuda:0'), tensor([22125,  7336, 44423,  3336,   376,  3843,
11335,   532, 34958,   406,
        49494,   569,   986], device='cuda:0'), tensor([   50,  8845,  2767,   309,
28082], device='cuda:0'), tensor([50241, 31376,    25,  3497,  3806,   383, 19175],
device='cuda:0'), tensor([ 3123,  1052,   377,   272,   555, 26849,   257, 24568, 10
115, 36309,
          390, 15095, 10094,   986], device='cuda:0'), tensor([ 2061,   262, 25003,
12248,    30,   351,   509,  6981,  6500],
       device='cuda:0'), tensor([39152,   390,  4881, 17364,   572,   355, 21166, 20
070,   329, 21933],
       device='cuda:0'), tensor([16157,   604,   400, 16134, 10185], device='cuda:
0'), tensor([16775,    25, 19100,  3050,    25, 17905,   422,   262, 42754],
       device='cuda:0'), tensor([47379, 39941,   351,  2561, 12880, 11252], device
='cuda:0'), tensor([    1,  2504,   338, 16013,     1,  4849,  1769, 16013, 22520,
12,
        15307, 42495], device='cuda:0'), tensor([14254,  5896], device='cuda:0'), te
nsor([32470, 19705,  1004,    65,  1313, 17906,   354,   317, 17001],
       device='cuda:0'), tensor([ 5962, 18720,    19,   983,   287,   257,   981,
357, 19419, 14604,
            8], device='cuda:0'), tensor([  72, 3987,  470], device='cuda:0'), tenso
r([49898,  4502,   532,   366,  8048,  2080,  5896,     1],
       device='cuda:0'), tensor([21868,  3691,   360,   967,   805], device='cuda:
0'), tensor([20238,  1374,  1675,  6869, 11152], device='cuda:0'), tensor([   35,  3
861,  7336, 44253,  6465,    56,  2538,  6226,  6684,    37,
         3228], device='cuda:0'), tensor([   1, 2061, 1002,    1,  532, 6416, 1204,
550, 2008,  983, 2128, 3048],
       device='cuda:0'), tensor([41173,     6,    51,   309, 15675], device='cuda:
0'), tensor([41762,   364,   911,  2096,  1550,   383,   350,   808,    75],
       device='cuda:0'), tensor([  38, 7197,  532, 7610, 1550, 7849, 7623], device
='cuda:0'), tensor([48412,    53,  8270, 16036,   532,  2365,  1315,   400],
       device='cuda:0'), tensor([   34,  3861,    57,    56,   347, 31949,  1546,  3
228],
       device='cuda:0'), tensor([31660,   163,   110,   240, 36365,   112,   163,
237,   254,   162,
          236,   231, 34460,   110,   165,   238,   113,   162,   110,   247],
       device='cuda:0'), tensor([45708,   805,   532,  7542,  5476,   361,   993,
319,  3899,  6612],
```

       device='cuda:0'), tensor([3351,   631, 10813,   775, 48038,   532, 5896], d
evice='cuda:0'), tensor([35, 20474, 44322, 9212, 16013, 5948, 1987,   400, 29
32, 7769,
         532, 2644], device='cuda:0'), tensor([18081,   88, 13840,   329, 2444,
290,   584, 2644],
       device='cuda:0'), tensor([24151, 8953,   656, 29420,   981, 36634], device
='cuda:0'), tensor([8264, 22707,   338,   371,   13, 5741, 5256, 1326,   88,
11,
        12655,   319, 8378,   286, 1675,   986], device='cuda:0'), tensor([15017,
31045, 3764,   860,   12,   24,   12,   940],
       device='cuda:0'), tensor([46751, 8407,   532, 12169, 39348], device='cuda:
0'), tensor([39, 2586,   329, 18509, 2586], device='cuda:0'), tensor([12442, 39
139, 9252,   532, 7922, 1248], device='cuda:0'), tensor([13256, 6612,   532,   3
83, 5896,   921, 12793], device='cuda:0'), tensor([51, 8254, 16483, 27433, 509
4, 21983], device='cuda:0'), tensor([2662,   38, 5181, 16860, 34017, 18448], devi
ce='cuda:0'), tensor([42,   437, 5557, 29618, 6056, 5250, 15626,   319, 7631,
986],
       device='cuda:0'), tensor([7376, 9259, 12569, 41527, 31772,   0], device
='cuda:0'), tensor([5962, 7755, 10682], device='cuda:0'), tensor([13448, 9659, 7
967, 27794, 5075], device='cuda:0'), tensor([2257, 2200, 2767, 39484, 2149,   5
13], device='cuda:0'), tensor([49757, 38923, 23624, 9986], device='cuda:0'), tensor
([48640, 1222, 44485, 15253], device='cuda:0'), tensor([5211,   407, 9870, 13118,
20051,   0, 3409, 19155,   986],
       device='cuda:0'), tensor([9915, 21566,   276, 2631,   292,   532,   314,
402, 12375, 39635,
         357, 10913,   986], device='cuda:0'), tensor([18102,   286, 8576,   337,
7211, 1039], device='cuda:0'), tensor([22973,   382,   532, 4280, 1098,   685, 27
977, 2149, 12576, 35507,
          60], device='cuda:0'), tensor([7206, 2885, 24027,   262, 1642,   286],
device='cuda:0'), tensor([3546, 2611, 3497, 632, 554, 532, 2520, 8511, 5521, 44
8,   0],
       device='cuda:0'), tensor([29907, 22233,   11, 1248, 2709, 1756, 8533,
532, 2253,   338,
        11853, 17388,   986], device='cuda:0'), tensor([10970, 9440, 6489, 3955,
15365, 7375, 3069, 6242],
       device='cuda:0'), tensor([27722,   12,   51, 1726,   262, 3000, 1303, 1
485,   25, 5059,
          13, 37727,   13, 25635,   13,   357,   701,   13,   775, 8471],
       device='cuda:0'), tensor([46827, 8314, 4849,   89, 7423], device='cuda:
0'), tensor([38,   859, 3876, 19147], device='cuda:0'), tensor([49174,   616, 1
097], device='cuda:0'), tensor([23812, 3406, 4942,   68,   25, 3738, 42722, 159
90, 1559, 1220,
        15585, 2644], device='cuda:0'), tensor([43, 9632,   406, 1590, 1802,
23, 18360, 5056,   265,   11,
         290,   350, 48809], device='cuda:0'), tensor([33, 31777,   338, 8407,
33233,   532,   220,   943, 7012, 1031,
          11, 31995,   257,   986], device='cuda:0'), tensor([14157, 5913], device
='cuda:0'), tensor([3855, 3406, 7623,   319, 7092, 23656,   0], device='cuda:
0'), tensor([2061,   262, 25003, 12248,   30,   351,   509, 6981, 6500],
       device='cuda:0'), tensor([4535, 7340, 6535,   56, 40342,   53, 25633,
402, 4663, 6561,
           0,   357, 1157,   14, 2078,   14, 2931,   12, 26276,   8],
       device='cuda:0'), tensor([50,   624,   286,   921,   532, 7257, 7336,
357, 28529, 7623,
           8], device='cuda:0'), tensor([45, 8874, 18694, 8874,   220, 3806,
2975, 7930,   400, 27752,

12873, 3717], device='cuda:0'), tensor([34,   372, 2645, 11768,   532,
34920,   770,   930, 6462, 8829,
29496,   357, 41275,   8], device='cuda:0'), tensor([3791, 32052,   350,
873,   286, 6997, 3442,   290,   49,   72,
986], device='cuda:0'), tensor([49, 4449, 7697,   25, 2011, 33869],
device='cuda:0'), tensor([13965, 40195, 49224, 5565, 42399, 31091, 3021,   532, 42
700, 2813,
2644], device='cuda:0'), tensor([33869,   338, 18881,   805,   338, 50125,
341], device='cuda:0'), tensor([464, 6785, 1114, 4687], device='cuda:0'), tensor([3
4720, 9936, 6006, 10067, 3180,   311, 6998, 24218, 8035, 13896],
device='cuda:0'), tensor([25433, 50133,   539, 9280], device='cuda:0'), tens
or([22797, 1869,   362], device='cuda:0'), tensor([464, 350, 414, 5172], device
='cuda:0'), tensor([21077,   25, 9164, 20051,   286,   968, 1971, 20206],
device='cuda:0'), tensor([7414,   789, 20745,   532, 18773, 24015,   291, 7
561,   311,   986],
device='cuda:0'), tensor([3109, 9124,   653, 5478,   25, 21167,   262, 5
905, 28089,   532,
28697], device='cuda:0')], 'query': ['What to buy the women in your life
-...', 'Re: @knitmeapony | Old Spice', 'Lindsay Lohan Needs Real Friends (Ep #1)',
'Rainy Days', "Letterman - Dave's Monologue Excerpt:...", 'So I met Miley for real
z!', 'Ann Marie Calhoun - "The Pretend...', 'Mega64: The Beatles Rock Band', 'Black
Ops: Free for All 1 & 2', 'Jumbo Jet vs People on the Beach', 'STAR TREK - Angry Vid
eo Game Nerd', 'Why do you watch the videos I ma...', 'There Will Be Bud', 'Actuació
n Insula Kampa, Pub Ágora 3.12.2010', 'The Adventures of Batman and Rob...', 'Float
Like A Butterfly', "The Cacaman - Ain't Got Much of...", 'Paladino threatens New Yor
k Post Editor', 'kaylee', "ITN: Japan radiation 'causes no concern'", 'The Hauntenin
g', 'Chicken!Chicken!', 'PALE BLUE DOT', 'Make a difference', 'In the car with James
May - Top Gear - BBC', 'History Will Be Made - Radulov/Arnott', 'Polish Police SCBA
- Chopper', 'IMAX in a basement', 'Ned 1 Den 0', 'BUTT MUSCLES & HEAVY BREATHING',
'Giant House Spider & COMMENT CONTEST 蜘蛛', 'imadethismistake - gravediggers...',
'Favre Struck By Football in a Groin Shot- Says...', 'Baby Bunny Grooming', '11 Reas
ons to Vote for Democrats in N...', 'BoondocksBootleg.com Exclusive: Stinkmeaner Rem
ix\'s Usher\'s "N', '"CONNECT" Regime Change, Negus World Order, Saba Saba, Narcy',
'Pokémon FireRed - Part 17', 'Style Icon Malaika Shares health Tips with her...', 'K
aty Perry talks "Peacock"', 'underwater rugby bratislava', 'Surfwise: The Amazing Tr
ue Odyssey of the Paskowitz Family', 'PATADON DE DRENTHE EN EL PECHO A UN J...', 'mo
ded', 'Teens Make Offensive T-Shirt - Should It Be Banned?', 'Huge Spanish Knocker
s!!! (8.31.09 - D...', 'BLOW JOB BRAZILIAN STYLE?', "Jake's Haircut", 'Modern Warfar
e 2 : Shotgun Sally Episode 1 - Search &', 'NIKE WRITE THE FUTURE - FULL LENGTH
V...', 'SWEET TALK', 'Terry Tate: Get Out The Vote', 'Le Anulan un Gol a Cristiano R
onaldo de globito...', 'What the WHAT?!? with KANDEE', 'Tour de France kicks off as
Armstrong pushes for comeback', 'July 4th parade!!!', 'Project:Report 2010: Reports
from the Winners', 'BAT FIGHT with Will Ferrell', '"That\'s Gay" Salutes Gay Reality
-Show Judges', 'Twitter Love', 'NBA Finals Lebron Etch A Sketch', 'First Cod4 game i
n a while (without commentary)', "iDon't", 'Victoria George - "Down With Love"', 'Ry
an vs Dorkman', 'Learn How To Moonwalk', 'DRAKE FREESTYLE SPOOF!!', '"What If" - Rea
l life had video game sound effects', "DON'T TREAD", 'Transformers Shorts On The Pro
wl', 'Glee - Dream On Music Video', 'FarmVille Podcast - Jan 15th', 'CRAZY BITCHE
S!!', '一粒水珠掉進鐵沙', 'Letterman - Queen Latifah on Michael Jackson', 'Screeching
Weasel - Love', 'DMG Dill Mill Gayye 24th August 09 -...', 'Healthy meals for studen
ts and other...', 'Girl falls into fountain while texting', "GEICO's R. Lee Ermey, a
ppearing on behalf of To...", 'San Bruno Fire 9-9-10', 'Jacob Golden - Zero Integrit
y', 'Homes for Gnomes', 'Super Luigi Galaxy - Episode 18', 'Michael Jackson - The Lo
ve You Save', 'Tiger Woods Makes Public Statement', 'OMG Cat watches Orchestra Fai
l', 'Kendrick Perkins flagrant foul on Mic...', 'Cheesy Ringtone Songs!', 'First Per
son Mario', 'Matt Allen Snowboarding 2005', 'STREET MUSIC 3', 'Moscow Subway Traffic

Jam', 'Emily & Henrietta', 'Do not Trust Profile Pictures! Samsun...', 'Black Eyed P
eas - I Gotta Feeling (FR...', 'House of 1000 Muppets', 'Paramore - Decode [OFFICIAL
VIDEO]', 'DEADLINE the making of', 'Imma Get It In - South Beach Workout!', "Taylor
Matthews, 18 auditions Dallas - America's Got Tale...", 'THE COMPLIMENTS COLLAB', 'A
uto-Tune the News #13: driving. stripping. swinging. (ft. Weez', 'Maria Does Salzbur
g', 'Grammar Nazis', 'Destroy my car', 'Know Your Meme: Antoine Dodson / Bed...', 'L
ogan Lacy 1008 lb Squat, and Puked', "Bollywood's Golden Triangle - Arbaaz, Salman
a...", 'North Carolina', 'Get Your Video on SportsCenter!', 'What the WHAT?!? with K
ANDEE', 'NAUGHTY ELEVATOR GIRLS! (11/28/09-269)', 'Sick of You - CAKE (Official Vide
o)', 'NOVIKOV Out road 66th Rally Poland 2009', 'Cheryl Cole - Promise This | Full
Radio Rip (HQ)', 'New Dirty Pics of Ms California andRi...', 'Rihanna: MyYouTube',
'NEW Toshiba Android Tablet Revealed - CES 2011...', "YouTube's Layman's Explanatio
n", 'The Quest For Space', 'DIRTY SHAME IS SRS BSNS!!!!', 'milkshake dance', 'Iron M
an 2', 'The Pity Card', 'Found: Lost Pictures of New York Blizzard', 'Flower Warfare
- Psychedelic Action S...', 'Expedition Africa: Meet the Explorers - Benedict'], 're
sponse': [' More\n\nBackground\n', ' Hothouse\n\nSubject:', ' Kristen Bell and her',
'" was my favorite "Community"', ' Free View in iTunes\n', ' Lyrics by Hva Toll', '"
January 8,', '). This Mesmorizer', ' Weapon DLC\n\nDecember 22', ' –\n\nPir', '\n\nY
es, you read', '\n\nI found', 's", and "Wra', '\n\nPico-Cola /', ' Free View in iTun
es\n\n', ' 5.69 574 5', '\n\n30.46', "'s Journal\n\n", '.com\n\nWanna Know', ' says
health ministry Read more', ' conjurers are a small', 'Every time you make a "re', '
FOR ADULT M', ' with your Ether.\n\n', ' Radio 4 and BBC the Magazine', ' Would Be T
he Mess Hallers -', ' Rather at Risk - Cubism', ' office.\n\nHe projected the', ' Ad
am 4 Wisconsin Rlw 5', '\n\nThough we do not recommend', '粉SpiderWarrior', 'this mu
st be your spark', ' After Being Robbed, Buddy', ' Adult Ballshops\n\nW', ' Between
now and November, more than', 'igga"!Our exclusive', ' Freeling, Double Doctors, Tra
nsfer', ' Pokémon FireRed - Part 18 Pokémon', ' Her 5 tips are day 1', ' with Fred A
rmisen. Bob', '.proofwatchstore.photos\n', '\n\nThis family', 'A NICE ONES', ' Choic
e telephone and tablet is now', '" on the police and a police', '\n\n\nMexican Resta
urant in', ' WEB GOING -', " is also Bryce's Pizza.", ' Rescue Tali Episode 1', ' Fr
ee View in iTunes', '\n\nBut0 BiS', ' Convention Unite (series)', '\n\nHilario Garci
a Ern', ' SOUNDS LIKE', '\n\nFormer champion Michael Johnson attacks', ' 5 West that
route was getting', ' Café (2011 to 2014)\n', ' and a huge crowd', ' (Both MTV and',
' for Liberals is not a new', ' (38) Hop', " started today, and it's", ': All Wars.
By displaying an', ' (Part 2) So the', ' listed name is now (10', ' Dropping The Pan
ties FAQ\n', ' Inaugural car', '. I could listen', ' ON GOLD\n\n', '180611-010', ' G
ame By Tom Harding Producer Getty', ', 2016\n\n', ' It may not be that', '的一翻&#xfffd;',
'\n\nPolitica - Elis', ' at First Sight 80 4081', ' Mae 45 landing from Dive n', ' m
eals for students and other...<|endoftext|>', ' boyfriend pic.twitter.', ' Free View
in iTunes', ' 20 6223 47', '\n\nLevel 8', ' module fuelling the Trade Route expansio
n', ': RAMONE launches underwater\n\n', ' (Intro Mix', ' on Child Sex Abuse I wante
d', ' in Print Mackenzie County,', '\n\nFree throws', 'Machine Download Download Fas
ter', ' fan: "This is one of', ' Dave Levin DUD', "'s eclectic pop/rock", '\n\nStopp
ing for sound', '. There is no', 'idioporn-', '\n\nDarkha - Fetish', '" was the four
th episode', '\n\nAlly Baby - Tur', ' the Middle East.', ' feat. AvaAngel\n', ' - 6/
13/', 'ORATION ACT, 1978', "o) Everything you'll", '? Deborah Seidl-W', "," one of t
he', ' with it."\n', ' Bens / End', " Sneakers, I'm", ' # #Indians Are', ' and Nebra
ska 0 12 17120', '\n\n"While we', ' PRINTES below', '\n\n7.', " Jones' debut on th
e", '/10\n\nNOV', ' | 53:14 Clip', ' 50 pages - Reportedly', 'BabyMon: Lord, how',
'\n\nThe Full Tos', ' for Vovk', ' Iswill heroes old', ' Mens Rights Extremism', ".
Yes it's delicious and the", " seafaring among the Captain Comet's", "'s version of
Limited to", ' November 10, 2013Added:', '\n\nHome Alone -', ' Paul-Goes\n\nTwo']}

In [31]:
```
batch.keys()
```

Out[31]: dict_keys(['input_ids', 'query', 'response'])

Compute Score

In [32]:
```
batch["query"]
```

Out[32]:
```
['What to buy the women in your life -...',
 'Re: @knitmeapony | Old Spice',
 'Lindsay Lohan Needs Real Friends (Ep #1)',
 'Rainy Days',
 "Letterman - Dave's Monologue Excerpt:...",
 'So I met Miley for realz!',
 'Ann Marie Calhoun - "The Pretend...',
 'Mega64: The Beatles Rock Band',
 'Black Ops: Free for All 1 & 2',
 'Jumbo Jet vs People on the Beach',
 'STAR TREK - Angry Video Game Nerd',
 'Why do you watch the videos I ma...',
 'There Will Be Bud',
 'Actuación Insula Kampa, Pub Ágora 3.12.2010',
 'The Adventures of Batman and Rob...',
 'Float Like A Butterfly',
 "The Cacaman - Ain't Got Much of...",
 'Paladino threatens New York Post Editor',
 'kaylee',
 "ITN: Japan radiation 'causes no concern'",
 'The Hauntening',
 'Chicken!Chicken!',
 'PALE BLUE DOT',
 'Make a difference',
 'In the car with James May - Top Gear - BBC',
 'History Will Be Made - Radulov/Arnott',
 'Polish Police SCBA - Chopper',
 'IMAX in a basement',
 'Ned 1 Den 0',
 'BUTT MUSCLES & HEAVY BREATHING',
 'Giant House Spider & COMMENT CONTEST  蜘蛛',
 'imadethismistake - gravediggers...',
 'Favre Struck By Football in a Groin Shot- Says...',
 'Baby Bunny Grooming',
 '11 Reasons to Vote for Democrats in N...',
 'BoondocksBootleg.com Exclusive: Stinkmeaner Remix\'s Usher\'s "N',
 '"CONNECT" Regime Change, Negus World Order, Saba Saba, Narcy',
 'Pokémon FireRed - Part 17',
 'Style Icon Malaika Shares health Tips with her...',
 'Katy Perry talks "Peacock"',
 'underwater rugby bratislava',
 'Surfwise: The Amazing True Odyssey of the Paskowitz Family',
 'PATADON DE DRENTHE EN EL PECHO A UN J...',
 'moded',
 'Teens Make Offensive T-Shirt - Should It Be Banned?',
 'Huge Spanish Knockers!!! (8.31.09 - D...',
 'BLOW JOB BRAZILIAN STYLE?',
 "Jake's Haircut",
 'Modern Warfare 2 : Shotgun Sally Episode 1 - Search &',
 'NIKE WRITE THE FUTURE - FULL LENGTH V...',
 'SWEET TALK',
 'Terry Tate: Get Out The Vote',
 'Le Anulan un Gol a Cristiano Ronaldo de globito...',
 'What the WHAT?!? with KANDEE',
 'Tour de France kicks off as Armstrong pushes for comeback',
 'July 4th parade!!!',
```

'Project:Report 2010: Reports from the Winners',
'BAT FIGHT with Will Ferrell',
'"That\'s Gay" Salutes Gay Reality-Show Judges',
'Twitter Love',
'NBA Finals Lebron Etch A Sketch',
'First Cod4 game in a while (without commentary)',
"iDon't",
'Victoria George - "Down With Love"',
'Ryan vs Dorkman',
'Learn How To Moonwalk',
'DRAKE FREESTYLE SPOOF!!',
'"What If" - Real life had video game sound effects',
"DON'T TREAD",
'Transformers Shorts On The Prowl',
'Glee - Dream On Music Video',
'FarmVille Podcast - Jan 15th',
'CRAZY BITCHES!!',
'一粒水珠掉進鐵沙',
'Letterman - Queen Latifah on Michael Jackson',
'Screeching Weasel - Love',
'DMG Dill Mill Gayye 24th August 09 -...',
'Healthy meals for students and other...',
'Girl falls into fountain while texting',
"GEICO's R. Lee Ermey, appearing on behalf of To...",
'San Bruno Fire 9-9-10',
'Jacob Golden - Zero Integrity',
'Homes for Gnomes',
'Super Luigi Galaxy - Episode 18',
'Michael Jackson - The Love You Save',
'Tiger Woods Makes Public Statement',
'OMG Cat watches Orchestra Fail',
'Kendrick Perkins flagrant foul on Mic...',
'Cheesy Ringtone Songs!',
'First Person Mario',
'Matt Allen Snowboarding 2005',
'STREET MUSIC 3',
'Moscow Subway Traffic Jam',
'Emily & Henrietta',
'Do not Trust Profile Pictures! Samsun...',
'Black Eyed Peas - I Gotta Feeling (FR...',
'House of 1000 Muppets',
'Paramore - Decode [OFFICIAL VIDEO]',
'DEADLINE the making of',
'Imma Get It In - South Beach Workout!',
"Taylor Matthews, 18 auditions Dallas - America's Got Tale...",
'THE COMPLIMENTS COLLAB',
'Auto-Tune the News #13: driving. stripping. swinging. (ft. Weez',
'Maria Does Salzburg',
'Grammar Nazis',
'Destroy my car',
'Know Your Meme: Antoine Dodson / Bed...',
'Logan Lacy 1008 lb Squat, and Puked',
"Bollywood's Golden Triangle - Arbaaz, Salman a...",
'North Carolina',
'Get Your Video on SportsCenter!',
'What the WHAT?!? with KANDEE',

```
 'NAUGHTY ELEVATOR GIRLS! (11/28/09-269)',
 'Sick of You - CAKE (Official Video)',
 'NOVIKOV  Out road 66th Rally Poland 2009',
 'Cheryl Cole - Promise This | Full Radio Rip (HQ)',
 'New Dirty Pics of Ms California andRi...',
 'Rihanna: MyYouTube',
 'NEW Toshiba Android Tablet Revealed - CES 2011...',
 "YouTube's Layman's Explanation",
 'The Quest For Space',
 'DIRTY SHAME IS SRS BSNS!!!!',
 'milkshake dance',
 'Iron Man 2',
 'The Pity Card',
 'Found: Lost Pictures of New York Blizzard',
 'Flower Warfare - Psychedelic Action S...',
 'Expedition Africa: Meet the Explorers - Benedict']
```

In [33]:
```
batch["response"]
```

Out[33]:
```
[' More\n\nBackground\n',
 ' Hothouse\n\nSubject:',
 ' Kristen Bell and her',
 '" was my favorite "Community"',
 ' Free View in iTunes\n',
 ' Lyrics by Hva Toll',
 '" January 8,',
 '). This Mesmorizer',
 ' Weapon DLC\n\nDecember 22',
 ' –\n\nPir',
 '\n\nYes, you read',
 '\n\nI found',
 's", and "Wra',
 '\n\nPico-Cola /',
 ' Free View in iTunes\n\n',
 ' 5.69 574 5',
 '\n\n30.46',
 "'s Journal\n\n",
 '.com\n\nWanna Know',
 ' says health ministry Read more',
 ' conjurers are a small',
 'Every time you make a "re',
 ' FOR ADULT M',
 ' with your Ether.\n\n',
 ' Radio 4 and BBC the Magazine',
 ' Would Be The Mess Hallers -',
 ' Rather at Risk - Cubism',
 ' office.\n\nHe projected the',
 ' Adam 4 Wisconsin Rlw 5',
 '\n\nThough we do not recommend',
 '粉SpiderWarrior',
 'this must be your spark',
 ' After Being Robbed, Buddy',
 ' Adult Ballshops\n\nW',
 ' Between now and November, more than',
 'igga"!Our exclusive',
 ' Freeling, Double Doctors, Transfer',
 ' Pokémon FireRed - Part 18 Pokémon',
 ' Her 5 tips are day 1',
 ' with Fred Armisen. Bob',
 '.proofwatchstore.photos\n',
 '\n\nThis family',
 'A NICE ONES',
 ' Choice telephone and tablet is now',
 '" on the police and a police',
 '\n\n\nMexican Restaurant in',
 ' WEB GOING -',
 " is also Bryce's Pizza.",
 ' Rescue Tali Episode 1',
 ' Free View in iTunes',
 '\n\nBut0 BiS',
 ' Convention Unite (series)',
 '\n\nHilario Garcia Ern',
 ' SOUNDS LIKE',
 '\n\nFormer champion Michael Johnson attacks',
 ' 5 West that route was getting',
```

```
' Café (2011 to 2014)\n',
' and a huge crowd',
' (Both MTV and',
' for Liberals is not a new',
' (38) Hop',
" started today, and it's",
': All Wars. By displaying an',
' (Part 2) So the',
' listed name is now (10',
' Dropping The Panties FAQ\n',
' Inaugural car',
'. I could listen',
' ON GOLD\n\n',
'180611-010',
' Game By Tom Harding Producer Getty',
', 2016\n\n',
' It may not be that',
'的一翻',
'\n\nPolitica - Elis',
' at First Sight 80 4081',
' Mae 45 landing from Dive n',
' meals for students and other...<|endoftext|>',
' boyfriend pic.twitter.',
' Free View in iTunes',
' 20 6223 47',
'\n\nLevel 8',
' module fuelling the Trade Route expansion',
': RAMONE launches underwater\n\n',
' (Intro Mix',
' on Child Sex Abuse I wanted',
' in Print Mackenzie County,',
'\n\nFree throws',
'Machine Download Download Faster',
' fan: "This is one of',
' Dave Levin DUD',
"'s eclectic pop/rock",
'\n\nStopping for sound',
'. There is no',
'idioporn-',
'\n\nDarkha - Fetish',
'" was the fourth episode',
'\n\nAlly Baby - Tur',
' the Middle East.',
' feat. AvaAngel\n',
' - 6/13/',
'ORATION ACT, 1978',
"o) Everything you'll",
'? Deborah Seidl-W',
'," one of the',
' with it."\n',
' Bens / End',
" Sneakers, I'm",
' # #Indians Are',
' and Nebraska 0 12 17120',
'\n\n"While we',
' PRINTES below',
```

```
'\n\n7.',
" Jones' debut on the",
'/10\n\nNOV',
' | 53:14 Clip',
' 50 pages - Reportedly',
'BabyMon: Lord, how',
'\n\nThe Full Tos',
' for Vovk',
' Iswill heroes old',
' Mens Rights Extremism',
". Yes it's delicious and the",
" seafaring among the Captain Comet's",
"'s version of Limited to",
' November 10, 2013Added:',
'\n\nHome Alone -',
' Paul-Goes\n\nTwo']
```

In [34]:
```
texts = [ q + r for q, r in zip(batch["query"], batch["response"]) ]
```

In [35]:
```
texts
```

Out[35]:
```
['What to buy the women in your life -... More\n\nBackground\n',
 'Re: @knitmeapony | Old Spice Hothouse\n\nSubject:',
 'Lindsay Lohan Needs Real Friends (Ep #1) Kristen Bell and her',
 'Rainy Days" was my favorite "Community"',
 "Letterman - Dave's Monologue Excerpt:... Free View in iTunes\n",
 'So I met Miley for realz! Lyrics by Hva Toll',
 'Ann Marie Calhoun - "The Pretend..." January 8,',
 'Mega64: The Beatles Rock Band). This Mesmorizer',
 'Black Ops: Free for All 1 & 2 Weapon DLC\n\nDecember 22',
 'Jumbo Jet vs People on the Beach –\n\nPir',
 'STAR TREK - Angry Video Game Nerd\n\nYes, you read',
 'Why do you watch the videos I ma...\n\nI found',
 'There Will Be Buds", and "Wra',
 'Actuación Insula Kampa, Pub Ágora 3.12.2010\n\nPico-Cola /',
 'The Adventures of Batman and Rob... Free View in iTunes\n\n',
 'Float Like A Butterfly 5.69 574 5',
 "The Cacaman - Ain't Got Much of...\n\n30.46",
 "Paladino threatens New York Post Editor's Journal\n\n",
 'kaylee.com\n\nWanna Know',
 "ITN: Japan radiation 'causes no concern' says health ministry Read more",
 'The Hauntening conjurers are a small',
 'Chicken!Chicken!Every time you make a "re',
 'PALE BLUE DOT FOR ADULT M',
 'Make a difference with your Ether.\n\n',
 'In the car with James May - Top Gear - BBC Radio 4 and BBC the Magazine',
 'History Will Be Made - Radulov/Arnott Would Be The Mess Hallers -',
 'Polish Police SCBA - Chopper Rather at Risk - Cubism',
 'IMAX in a basement office.\n\nHe projected the',
 'Ned 1 Den 0 Adam 4 Wisconsin Rlw 5',
 'BUTT MUSCLES & HEAVY BREATHING\n\nThough we do not recommend',
 'Giant House Spider & COMMENT CONTEST  蜘蛛粉SpiderWarrior',
 'imadethismistake - gravediggers...this must be your spark',
 'Favre Struck By Football in a Groin Shot- Says... After Being Robbed, Buddy',
 'Baby Bunny Grooming Adult Ballshops\n\nW',
 '11 Reasons to Vote for Democrats in N... Between now and November, more than',
 'BoondocksBootleg.com Exclusive: Stinkmeaner Remix\'s Usher\'s "Nigga"!Our exclus
ive',
 '"CONNECT" Regime Change, Negus World Order, Saba Saba, Narcy Freeling, Double Do
ctors, Transfer',
 'Pokémon FireRed - Part 17 Pokémon FireRed - Part 18 Pokémon',
 'Style Icon Malaika Shares health Tips with her... Her 5 tips are day 1',
 'Katy Perry talks "Peacock" with Fred Armisen. Bob',
 'underwater rugby bratislava.proofwatchstore.photos\n',
 'Surfwise: The Amazing True Odyssey of the Paskowitz Family\n\nThis family',
 'PATADON DE DRENTHE EN EL PECHO A UN J...A NICE ONES',
 'moded Choice telephone and tablet is now',
 'Teens Make Offensive T-Shirt - Should It Be Banned?" on the police and a polic
e',
 'Huge Spanish Knockers!!! (8.31.09 - D...\n\n\nMexican Restaurant in',
 'BLOW JOB BRAZILIAN STYLE? WEB GOING -',
 "Jake's Haircut is also Bryce's Pizza.",
 'Modern Warfare 2 : Shotgun Sally Episode 1 - Search & Rescue Tali Episode 1',
 'NIKE WRITE THE FUTURE - FULL LENGTH V... Free View in iTunes',
 'SWEET TALK\n\nBut0 BiS',
 'Terry Tate: Get Out The Vote Convention Unite (series)',
 'Le Anulan un Gol a Cristiano Ronaldo de globito...\n\nHilario Garcia Ern',
```

```
 'What the WHAT?!? with KANDEE SOUNDS LIKE',
 'Tour de France kicks off as Armstrong pushes for comeback\n\nFormer champion Mic
hael Johnson attacks',
 'July 4th parade!!! 5 West that route was getting',
 'Project:Report 2010: Reports from the Winners Café (2011 to 2014)\n',
 'BAT FIGHT with Will Ferrell and a huge crowd',
 '"That\'s Gay" Salutes Gay Reality-Show Judges (Both MTV and',
 'Twitter Love for Liberals is not a new',
 'NBA Finals Lebron Etch A Sketch (38) Hop',
 "First Cod4 game in a while (without commentary) started today, and it's",
 "iDon't: All Wars. By displaying an",
 'Victoria George - "Down With Love" (Part 2) So the',
 'Ryan vs Dorkman listed name is now (10',
 'Learn How To Moonwalk Dropping The Panties FAQ\n',
 'DRAKE FREESTYLE SPOOF!! Inaugural car',
 '"What If" - Real life had video game sound effects. I could listen',
 "DON'T TREAD ON GOLD\n\n",
 'Transformers Shorts On The Prowl180611-010',
 'Glee - Dream On Music Video Game By Tom Harding Producer Getty',
 'FarmVille Podcast - Jan 15th, 2016\n\n',
 'CRAZY BITCHES!! It may not be that',
 '一粒水珠掉進鐵沙的一翻',
 'Letterman - Queen Latifah on Michael Jackson\n\nPolitica - Elis',
 'Screeching Weasel - Love at First Sight 80 4081',
 'DMG Dill Mill Gayye 24th August 09 -... Mae 45 landing from Dive n',
 'Healthy meals for students and other... meals for students and other...<|endofte
xt|>',
 'Girl falls into fountain while texting boyfriend pic.twitter.',
 "GEICO's R. Lee Ermey, appearing on behalf of To... Free View in iTunes",
 'San Bruno Fire 9-9-10 20 6223 47',
 'Jacob Golden - Zero Integrity\n\nLevel 8',
 'Homes for Gnomes module fuelling the Trade Route expansion',
 'Super Luigi Galaxy - Episode 18: RAMONE launches underwater\n\n',
 'Michael Jackson - The Love You Save (Intro Mix',
 'Tiger Woods Makes Public Statement on Child Sex Abuse I wanted',
 'OMG Cat watches Orchestra Fail in Print Mackenzie County,',
 'Kendrick Perkins flagrant foul on Mic...\n\nFree throws',
 'Cheesy Ringtone Songs!Machine Download Download Faster',
 'First Person Mario fan: "This is one of',
 'Matt Allen Snowboarding 2005 Dave Levin DUD',
 "STREET MUSIC 3's eclectic pop/rock",
 'Moscow Subway Traffic Jam\n\nStopping for sound',
 'Emily & Henrietta. There is no',
 'Do not Trust Profile Pictures! Samsun...idioporn-',
 'Black Eyed Peas - I Gotta Feeling (FR...\n\nDarkha - Fetish',
 'House of 1000 Muppets" was the fourth episode',
 'Paramore - Decode [OFFICIAL VIDEO]\n\nAlly Baby - Tur',
 'DEADLINE the making of the Middle East.',
 'Imma Get It In - South Beach Workout! feat. AvaAngel\n',
 "Taylor Matthews, 18 auditions Dallas - America's Got Tale... - 6/13/",
 'THE COMPLIMENTS COLLABORATION ACT, 1978',
 "Auto-Tune the News #13: driving. stripping. swinging. (ft. Weezo) Everything yo
u'll",
 'Maria Does Salzburg? Deborah Seidl-W',
 'Grammar Nazis," one of the',
 'Destroy my car with it."\n',
```

```
'Know Your Meme: Antoine Dodson / Bed... Bens / End',
"Logan Lacy 1008 lb Squat, and Puked Sneakers, I'm",
"Bollywood's Golden Triangle -  Arbaaz, Salman a... # #Indians Are",
'North Carolina and Nebraska 0 12 17120',
'Get Your Video on SportsCenter!\n\n"While we',
'What the WHAT?!? with KANDEE PRINTES below',
'NAUGHTY ELEVATOR GIRLS! (11/28/09-269)\n\n7.',
"Sick of You - CAKE (Official Video) Jones' debut on the",
'NOVIKOV  Out road 66th Rally Poland 2009/10\n\nNOV',
'Cheryl Cole - Promise This | Full Radio Rip (HQ) | 53:14 Clip',
'New Dirty Pics of Ms California andRi... 50 pages - Reportedly',
'Rihanna: MyYouTubeBabyMon: Lord, how',
'NEW Toshiba Android Tablet Revealed - CES 2011...\n\nThe Full Tos',
"YouTube's Layman's Explanation for Vovk",
'The Quest For Space Iswill heroes old',
'DIRTY SHAME IS SRS BSNS!!!! Mens Rights Extremism',
"milkshake dance. Yes it's delicious and the",
"Iron Man 2 seafaring among the Captain Comet's",
"The Pity Card's version of Limited to",
'Found: Lost Pictures of New York Blizzard November 10, 2013Added:',
'Flower Warfare - Psychedelic Action S...\n\nHome Alone -',
'Expedition Africa: Meet the Explorers - Benedict Paul-Goes\n\nTwo']
```

In [36]:
```
pipe_outputs = sentiment_pipe(texts, **sent_kwargs)
pipe_outputs
```

Out[36]:

```
[[{'label': 'NOT', 'score': 2.490553617477417},
  {'label': 'CLICKBAIT', 'score': -2.9494712352752686}],
 [{'label': 'NOT', 'score': 0.8059258460998535},
  {'label': 'CLICKBAIT', 'score': -1.302541971206665}],
 [{'label': 'NOT', 'score': -0.4256044924259186},
  {'label': 'CLICKBAIT', 'score': -0.1774040162563324}],
 [{'label': 'NOT', 'score': 1.0809476375579834},
  {'label': 'CLICKBAIT', 'score': -1.4847525358200073}],
 [{'label': 'NOT', 'score': 3.8315069675445557},
  {'label': 'CLICKBAIT', 'score': -4.299412727355957}],
 [{'label': 'NOT', 'score': 2.443516492843628},
  {'label': 'CLICKBAIT', 'score': -3.1101608276367188}],
 [{'label': 'NOT', 'score': 2.957875967025757},
  {'label': 'CLICKBAIT', 'score': -3.311002492904663}],
 [{'label': 'NOT', 'score': 3.8029301166534424},
  {'label': 'CLICKBAIT', 'score': -4.127135276794434}],
 [{'label': 'NOT', 'score': 2.576737880706787},
  {'label': 'CLICKBAIT', 'score': -3.088524580001831}],
 [{'label': 'NOT', 'score': 3.8692023754119873},
  {'label': 'CLICKBAIT', 'score': -4.296860694885254}],
 [{'label': 'NOT', 'score': 0.6013877391815186},
  {'label': 'CLICKBAIT', 'score': -1.058555245399475}],
 [{'label': 'NOT', 'score': 3.553706645965576},
  {'label': 'CLICKBAIT', 'score': -3.7888102531433105}],
 [{'label': 'NOT', 'score': 3.1472647190093994},
  {'label': 'CLICKBAIT', 'score': -3.829800844192505}],
 [{'label': 'NOT', 'score': 3.699019193649292},
  {'label': 'CLICKBAIT', 'score': -4.202468395233154}],
 [{'label': 'NOT', 'score': 3.6021194458007812},
  {'label': 'CLICKBAIT', 'score': -3.964285373687744}],
 [{'label': 'NOT', 'score': 1.2105509042739868},
  {'label': 'CLICKBAIT', 'score': -1.1185709238052368}],
 [{'label': 'NOT', 'score': 3.1507885456085205},
  {'label': 'CLICKBAIT', 'score': -3.839857339859009}],
 [{'label': 'NOT', 'score': 4.275155544281006},
  {'label': 'CLICKBAIT', 'score': -4.536465644836426}],
 [{'label': 'NOT', 'score': 2.9418599605560303},
  {'label': 'CLICKBAIT', 'score': -3.2125041484832764}],
 [{'label': 'NOT', 'score': 4.291418075561523},
  {'label': 'CLICKBAIT', 'score': -4.494740009307861}],
 [{'label': 'NOT', 'score': 3.3783512115478516},
  {'label': 'CLICKBAIT', 'score': -3.8024017810821533}],
 [{'label': 'NOT', 'score': -2.434818983078003},
  {'label': 'CLICKBAIT', 'score': 2.6146416664123535}],
 [{'label': 'NOT', 'score': 2.834986925125122},
  {'label': 'CLICKBAIT', 'score': -3.188957452774048}],
 [{'label': 'NOT', 'score': 3.488293409347534},
  {'label': 'CLICKBAIT', 'score': -3.933056116104126}],
 [{'label': 'NOT', 'score': 3.7181427478790283},
  {'label': 'CLICKBAIT', 'score': -4.289842128753662}],
 [{'label': 'NOT', 'score': 0.6020993590354919},
  {'label': 'CLICKBAIT', 'score': -0.46480026841163635}],
 [{'label': 'NOT', 'score': 4.792882919311523},
  {'label': 'CLICKBAIT', 'score': -5.032668113708496}],
 [{'label': 'NOT', 'score': 2.167309284210205},
  {'label': 'CLICKBAIT', 'score': -2.4056763648986816}],
```

```
[{'label': 'NOT', 'score': 3.939134359359741},
 {'label': 'CLICKBAIT', 'score': -4.113234996795654}],
[{'label': 'NOT', 'score': -3.7135655879974365},
 {'label': 'CLICKBAIT', 'score': 4.077047824859619}],
[{'label': 'NOT', 'score': 2.6815121173858643},
 {'label': 'CLICKBAIT', 'score': -3.264343738555908}],
[{'label': 'NOT', 'score': 2.538802146911621},
 {'label': 'CLICKBAIT', 'score': -2.701378107070923}],
[{'label': 'NOT', 'score': 4.070515155792236},
 {'label': 'CLICKBAIT', 'score': -4.401711940765381}],
[{'label': 'NOT', 'score': 3.692721366882324},
 {'label': 'CLICKBAIT', 'score': -3.960734128952026 4}],
[{'label': 'NOT', 'score': -2.282127618789673},
 {'label': 'CLICKBAIT', 'score': 2.688727378845215}],
[{'label': 'NOT', 'score': -2.1414239406585693},
 {'label': 'CLICKBAIT', 'score': 2.38420033454895}],
[{'label': 'NOT', 'score': 3.6882803440093994},
 {'label': 'CLICKBAIT', 'score': -4.063819408416748}],
[{'label': 'NOT', 'score': 2.1010425090789795},
 {'label': 'CLICKBAIT', 'score': -2.1604177951812744}],
[{'label': 'NOT', 'score': -2.7447640895843506},
 {'label': 'CLICKBAIT', 'score': 3.2921204566955566}],
[{'label': 'NOT', 'score': 1.875684142112732},
 {'label': 'CLICKBAIT', 'score': -2.6237428188323975}],
[{'label': 'NOT', 'score': 3.901437282562256},
 {'label': 'CLICKBAIT', 'score': -4.069407939910889}],
[{'label': 'NOT', 'score': 3.5164177417755127},
 {'label': 'CLICKBAIT', 'score': -3.801755428314209}],
[{'label': 'NOT', 'score': 1.0390874147415161},
 {'label': 'CLICKBAIT', 'score': -1.4327014684677124}],
[{'label': 'NOT', 'score': 4.069339275360107},
 {'label': 'CLICKBAIT', 'score': -4.284747123718262}],
[{'label': 'NOT', 'score': -2.2419285774230957},
 {'label': 'CLICKBAIT', 'score': 2.5920650959014893}],
[{'label': 'NOT', 'score': 3.245828151702881},
 {'label': 'CLICKBAIT', 'score': -3.680583953857422}],
[{'label': 'NOT', 'score': 3.921677827835083},
 {'label': 'CLICKBAIT', 'score': -4.0511956214904785}],
[{'label': 'NOT', 'score': -0.7552233338356018},
 {'label': 'CLICKBAIT', 'score': 0.5039904713630676}],
[{'label': 'NOT', 'score': 2.8198940753936768},
 {'label': 'CLICKBAIT', 'score': -3.476113796234131}],
[{'label': 'NOT', 'score': 3.0744428634643555},
 {'label': 'CLICKBAIT', 'score': -3.425720453262329}],
[{'label': 'NOT', 'score': 1.3426051139831543},
 {'label': 'CLICKBAIT', 'score': -1.795684814453125}],
[{'label': 'NOT', 'score': 0.8154745101928711},
 {'label': 'CLICKBAIT', 'score': -1.036782860755920 4}],
[{'label': 'NOT', 'score': 4.063251972198486},
 {'label': 'CLICKBAIT', 'score': -4.314174175262451}],
[{'label': 'NOT', 'score': 1.422480583190918},
 {'label': 'CLICKBAIT', 'score': -1.7292962074279785}],
[{'label': 'NOT', 'score': 4.470278263092041},
 {'label': 'CLICKBAIT', 'score': -4.723553657531738}],
[{'label': 'NOT', 'score': 2.5562009811401367},
 {'label': 'CLICKBAIT', 'score': -3.054701805114746}],
```

```
[{'label': 'NOT', 'score': 4.266685962677002},
 {'label': 'CLICKBAIT', 'score': -4.594472885131836}],
[{'label': 'NOT', 'score': 2.7003872394561768},
 {'label': 'CLICKBAIT', 'score': -3.2071335315704346}],
[{'label': 'NOT', 'score': 0.76141756772995},
 {'label': 'CLICKBAIT', 'score': -1.338637351989746}],
[{'label': 'NOT', 'score': 3.3885154724121094},
 {'label': 'CLICKBAIT', 'score': -3.7893943786621094}],
[{'label': 'NOT', 'score': 1.7868406772613525},
 {'label': 'CLICKBAIT', 'score': -1.9192849397659302}],
[{'label': 'NOT', 'score': -3.8428735733032227},
 {'label': 'CLICKBAIT', 'score': 3.8775746822357178}],
[{'label': 'NOT', 'score': 2.591484308242798},
 {'label': 'CLICKBAIT', 'score': -3.1586906909942627}],
[{'label': 'NOT', 'score': 2.7233364582061768},
 {'label': 'CLICKBAIT', 'score': -3.2332708835601807}],
[{'label': 'NOT', 'score': -0.8822117447853088},
 {'label': 'CLICKBAIT', 'score': 1.0059884786605835}],
[{'label': 'NOT', 'score': -2.8548479080200195},
 {'label': 'CLICKBAIT', 'score': 3.4044647216796875}],
[{'label': 'NOT', 'score': 2.037306070327759},
 {'label': 'CLICKBAIT', 'score': -2.2328457832336426}],
[{'label': 'NOT', 'score': 1.963986873626709},
 {'label': 'CLICKBAIT', 'score': -2.3579630851745605}],
[{'label': 'NOT', 'score': 0.056641172617673874},
 {'label': 'CLICKBAIT', 'score': -0.048549942672252655}],
[{'label': 'NOT', 'score': -1.8505007028579712},
 {'label': 'CLICKBAIT', 'score': 2.0630438327789307}],
[{'label': 'NOT', 'score': 3.135908842086792},
 {'label': 'CLICKBAIT', 'score': -3.8392953872680664}],
[{'label': 'NOT', 'score': 2.0994410514831543},
 {'label': 'CLICKBAIT', 'score': -2.551079750061035}],
[{'label': 'NOT', 'score': -3.532059669494629},
 {'label': 'CLICKBAIT', 'score': 3.8347556591033936}],
[{'label': 'NOT', 'score': 2.508687734603882},
 {'label': 'CLICKBAIT', 'score': -3.1806700229644775}],
[{'label': 'NOT', 'score': 4.54910135269165},
 {'label': 'CLICKBAIT', 'score': -4.9480977058410645}],
[{'label': 'NOT', 'score': 3.708211660385132},
 {'label': 'CLICKBAIT', 'score': -3.9913742542266846}],
[{'label': 'NOT', 'score': 4.401367664337158},
 {'label': 'CLICKBAIT', 'score': -4.69659423828125}],
[{'label': 'NOT', 'score': 3.20281982421875},
 {'label': 'CLICKBAIT', 'score': -3.5354604721069336}],
[{'label': 'NOT', 'score': -3.3062362670898438},
 {'label': 'CLICKBAIT', 'score': 3.5009984970092773}],
[{'label': 'NOT', 'score': 4.109504222869873},
 {'label': 'CLICKBAIT', 'score': -4.501187324523926}],
[{'label': 'NOT', 'score': 3.7212514877319336},
 {'label': 'CLICKBAIT', 'score': -4.136507511138916}],
[{'label': 'NOT', 'score': 3.8700175285339355},
 {'label': 'CLICKBAIT', 'score': -4.186642646789551}],
[{'label': 'NOT', 'score': 4.387486934661865},
 {'label': 'CLICKBAIT', 'score': -4.752230167388916}],
[{'label': 'NOT', 'score': 0.9743362069129944},
 {'label': 'CLICKBAIT', 'score': -1.5393351316452026}],
```

```
[{'label': 'NOT', 'score': 3.182849645614624},
 {'label': 'CLICKBAIT', 'score': -3.863825798034668}],
[{'label': 'NOT', 'score': 2.379321336746216},
 {'label': 'CLICKBAIT', 'score': -2.548948049545288}],
[{'label': 'NOT', 'score': 3.5699546337127686},
 {'label': 'CLICKBAIT', 'score': -3.938143491744995}],
[{'label': 'NOT', 'score': 3.3674192428588867},
 {'label': 'CLICKBAIT', 'score': -3.626559019088745}],
[{'label': 'NOT', 'score': 3.837820053100586},
 {'label': 'CLICKBAIT', 'score': -4.339358329772949}],
[{'label': 'NOT', 'score': -4.489175319671631},
 {'label': 'CLICKBAIT', 'score': 4.417422771453857}],
[{'label': 'NOT', 'score': 4.084190368652344},
 {'label': 'CLICKBAIT', 'score': -4.349583148956299}],
[{'label': 'NOT', 'score': 1.5588675737380981},
 {'label': 'CLICKBAIT', 'score': -2.238907814025879}],
[{'label': 'NOT', 'score': 4.06973123550415},
 {'label': 'CLICKBAIT', 'score': -4.4356842041015625}],
[{'label': 'NOT', 'score': -0.8114478588104248},
 {'label': 'CLICKBAIT', 'score': 1.1657898426055908}],
[{'label': 'NOT', 'score': 2.558680772781372},
 {'label': 'CLICKBAIT', 'score': -2.7018508911132812}],
[{'label': 'NOT', 'score': 3.260246992111206},
 {'label': 'CLICKBAIT', 'score': -3.9278066158294678}],
[{'label': 'NOT', 'score': 2.315066337585449},
 {'label': 'CLICKBAIT', 'score': -3.0943102836608887}],
[{'label': 'NOT', 'score': 3.652923345565796},
 {'label': 'CLICKBAIT', 'score': -4.333667755126953}],
[{'label': 'NOT', 'score': 4.050889492034912},
 {'label': 'CLICKBAIT', 'score': -4.347410202026367}],
[{'label': 'NOT', 'score': 2.9257164001464844},
 {'label': 'CLICKBAIT', 'score': -3.7229394912719727}],
[{'label': 'NOT', 'score': 3.9698567390441895},
 {'label': 'CLICKBAIT', 'score': -4.289936542510986}],
[{'label': 'NOT', 'score': 3.4455385208129883},
 {'label': 'CLICKBAIT', 'score': -3.8097894191741943}],
[{'label': 'NOT', 'score': 1.5241469144821167},
 {'label': 'CLICKBAIT', 'score': -1.9235386848449707}],
[{'label': 'NOT', 'score': 3.078164577484131},
 {'label': 'CLICKBAIT', 'score': -3.326512575149536}],
[{'label': 'NOT', 'score': 2.246835231781006},
 {'label': 'CLICKBAIT', 'score': -2.856437921524048}],
[{'label': 'NOT', 'score': -0.23542283475399017},
 {'label': 'CLICKBAIT', 'score': 0.27194687724113464}],
[{'label': 'NOT', 'score': -0.35999277234077454},
 {'label': 'CLICKBAIT', 'score': -0.04667913168668747}],
[{'label': 'NOT', 'score': 2.7356457710266113},
 {'label': 'CLICKBAIT', 'score': -3.141009569168091}],
[{'label': 'NOT', 'score': -3.239875078201294},
 {'label': 'CLICKBAIT', 'score': 3.5772030353546143}],
[{'label': 'NOT', 'score': 3.990867853164673},
 {'label': 'CLICKBAIT', 'score': -4.538613796234131}],
[{'label': 'NOT', 'score': 3.6158077716827393},
 {'label': 'CLICKBAIT', 'score': -3.947270154953003}],
[{'label': 'NOT', 'score': -0.31581810116767883},
 {'label': 'CLICKBAIT', 'score': 1.167500376701355}],
```

```
[{'label': 'NOT', 'score': 1.6431785821914673},
 {'label': 'CLICKBAIT', 'score': -1.4297008514404297}],
[{'label': 'NOT', 'score': 3.457759380340576},
 {'label': 'CLICKBAIT', 'score': -4.107631683349609}],
[{'label': 'NOT', 'score': 4.284034252166748},
 {'label': 'CLICKBAIT', 'score': -4.5322794914245605}],
[{'label': 'NOT', 'score': 3.162745237350464},
 {'label': 'CLICKBAIT', 'score': -3.675973653793335}],
[{'label': 'NOT', 'score': 2.546597957611084},
 {'label': 'CLICKBAIT', 'score': -3.086427688598633}],
[{'label': 'NOT', 'score': -2.1174662113189697},
 {'label': 'CLICKBAIT', 'score': 2.5836117267608643}],
[{'label': 'NOT', 'score': 3.34889554977417},
 {'label': 'CLICKBAIT', 'score': -3.961402654647827}],
[{'label': 'NOT', 'score': -3.423415422439575},
 {'label': 'CLICKBAIT', 'score': 3.5565338134765625}],
[{'label': 'NOT', 'score': 2.876871347427368},
 {'label': 'CLICKBAIT', 'score': -3.2152493000030518}],
[{'label': 'NOT', 'score': -2.4657669067382812},
 {'label': 'CLICKBAIT', 'score': 3.037748098373413}],
[{'label': 'NOT', 'score': -1.8238465785980225},
 {'label': 'CLICKBAIT', 'score': 2.2414450645446777}],
[{'label': 'NOT', 'score': 3.632723808288574},
 {'label': 'CLICKBAIT', 'score': -4.050164222717285}],
[{'label': 'NOT', 'score': 3.204392910003662},
 {'label': 'CLICKBAIT', 'score': -3.6643264293670654}],
[{'label': 'NOT', 'score': 3.9514124393463135},
 {'label': 'CLICKBAIT', 'score': -4.340408802032471}],
[{'label': 'NOT', 'score': 3.3821685314178467},
 {'label': 'CLICKBAIT', 'score': -3.9338936805725098}],
[{'label': 'NOT', 'score': 4.545583248138428},
 {'label': 'CLICKBAIT', 'score': -4.856073379516602}]]
```

In [37]:
```
rewards = [ torch.tensor(output[1]["score"]) for output in pipe_outputs]
rewards
```

Out[37]:
```
[tensor(-2.9495),
 tensor(-1.3025),
 tensor(-0.1774),
 tensor(-1.4848),
 tensor(-4.2994),
 tensor(-3.1102),
 tensor(-3.3110),
 tensor(-4.1271),
 tensor(-3.0885),
 tensor(-4.2969),
 tensor(-1.0586),
 tensor(-3.7888),
 tensor(-3.8298),
 tensor(-4.2025),
 tensor(-3.9643),
 tensor(-1.1186),
 tensor(-3.8399),
 tensor(-4.5365),
 tensor(-3.2125),
 tensor(-4.4947),
 tensor(-3.8024),
 tensor(2.6146),
 tensor(-3.1890),
 tensor(-3.9331),
 tensor(-4.2898),
 tensor(-0.4648),
 tensor(-5.0327),
 tensor(-2.4057),
 tensor(-4.1132),
 tensor(4.0770),
 tensor(-3.2643),
 tensor(-2.7014),
 tensor(-4.4017),
 tensor(-3.9607),
 tensor(2.6887),
 tensor(2.3842),
 tensor(-4.0638),
 tensor(-2.1604),
 tensor(3.2921),
 tensor(-2.6237),
 tensor(-4.0694),
 tensor(-3.8018),
 tensor(-1.4327),
 tensor(-4.2847),
 tensor(2.5921),
 tensor(-3.6806),
 tensor(-4.0512),
 tensor(0.5040),
 tensor(-3.4761),
 tensor(-3.4257),
 tensor(-1.7957),
 tensor(-1.0368),
 tensor(-4.3142),
 tensor(-1.7293),
 tensor(-4.7236),
 tensor(-3.0547),
```

```
tensor(-4.5945),
tensor(-3.2071),
tensor(-1.3386),
tensor(-3.7894),
tensor(-1.9193),
tensor(3.8776),
tensor(-3.1587),
tensor(-3.2333),
tensor(1.0060),
tensor(3.4045),
tensor(-2.2328),
tensor(-2.3580),
tensor(-0.0485),
tensor(2.0630),
tensor(-3.8393),
tensor(-2.5511),
tensor(3.8348),
tensor(-3.1807),
tensor(-4.9481),
tensor(-3.9914),
tensor(-4.6966),
tensor(-3.5355),
tensor(3.5010),
tensor(-4.5012),
tensor(-4.1365),
tensor(-4.1866),
tensor(-4.7522),
tensor(-1.5393),
tensor(-3.8638),
tensor(-2.5489),
tensor(-3.9381),
tensor(-3.6266),
tensor(-4.3394),
tensor(4.4174),
tensor(-4.3496),
tensor(-2.2389),
tensor(-4.4357),
tensor(1.1658),
tensor(-2.7019),
tensor(-3.9278),
tensor(-3.0943),
tensor(-4.3337),
tensor(-4.3474),
tensor(-3.7229),
tensor(-4.2899),
tensor(-3.8098),
tensor(-1.9235),
tensor(-3.3265),
tensor(-2.8564),
tensor(0.2719),
tensor(-0.0467),
tensor(-3.1410),
tensor(3.5772),
tensor(-4.5386),
tensor(-3.9473),
tensor(1.1675),
```

```
tensor(-1.4297),
tensor(-4.1076),
tensor(-4.5323),
tensor(-3.6760),
tensor(-3.0864),
tensor(2.5836),
tensor(-3.9614),
tensor(3.5565),
tensor(-3.2152),
tensor(3.0377),
tensor(2.2414),
tensor(-4.0502),
tensor(-3.6643),
tensor(-4.3404),
tensor(-3.9339),
tensor(-4.8561)]
```

In [38]:
```
len(rewards)
```

Out[38]: 128

In [39]:
```
for epoch, batch in tqdm(enumerate(ppo_trainer.dataloader)):
    query_tensors = batch["input_ids"]
    print(epoch)

    #### Get response from gpt2
    response_tensors = []
    for query in query_tensors:
        gen_len                              = output_length_sampler()
        generation_kwargs["max_new_tokens"] = gen_len
        response                             = ppo_trainer.generate(query, **generat
        response_tensors.append( response.squeeze()[-gen_len:] )
    batch["response"] = [ tokenizer.decode(r.squeeze()) for r in response_tensors ]

    #### Compute sentiment score
    texts = [q + r for q, r in zip(batch["query"], batch["response"])]
    pipe_outputs = sentiment_pipe(texts, **sent_kwargs)
    rewards = [ torch.tensor(output[1]["score"]) for output in pipe_outputs]

    #### Run PPO step
    stats = ppo_trainer.step(
                    query_tensors,
                    response_tensors,
                    rewards
    )
    ppo_trainer.log_stats(stats, batch, rewards)
```

```
0it [00:00, ?it/s]
0
1it [02:11, 131.94s/it]
1
2it [05:03, 155.05s/it]
2
3it [07:55, 162.91s/it]
```

```
3
4it [10:47, 166.55s/it]
4
5it [13:34, 166.78s/it]
5
6it [16:28, 169.03s/it]
6
7it [19:20, 170.06s/it]
7
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipelines\base.py:1123: UserWarning: You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency please use a dataset
  warnings.warn(
8it [22:12, 170.59s/it]
8
9it [25:05, 171.40s/it]
9
10it [27:56, 171.49s/it]
10
11it [30:49, 171.80s/it]
11
12it [33:32, 169.29s/it]
12
13it [36:25, 170.16s/it]
13
14it [39:18, 171.14s/it]
14
15it [42:11, 171.59s/it]
15
16it [45:04, 172.04s/it]
16
17it [47:57, 172.43s/it]
17
18it [50:50, 172.68s/it]
18
19it [53:44, 172.87s/it]
19
20it [56:38, 173.33s/it]
20
21it [59:31, 173.23s/it]
21
22it [1:02:16, 170.72s/it]
22
23it [1:05:03, 169.64s/it]
23
24it [1:07:51, 169.03s/it]
24
25it [1:10:38, 168.45s/it]
25
```

```
26it [1:13:26, 168.33s/it]
26
27it [1:16:14, 168.42s/it]
27
28it [1:19:01, 167.95s/it]
28
29it [1:21:49, 167.98s/it]
29
30it [1:24:37, 167.82s/it]
30
31it [1:27:30, 169.33s/it]
31
32it [1:30:22, 170.25s/it]
32
33it [1:33:14, 170.82s/it]
33
34it [1:36:08, 171.60s/it]
34
35it [1:39:01, 172.03s/it]
35
36it [1:41:54, 172.41s/it]
36
37it [1:44:47, 172.57s/it]
37
38it [1:47:36, 171.50s/it]
38
39it [1:50:22, 170.03s/it]
39
40it [1:53:16, 171.11s/it]
40
41it [1:56:03, 169.97s/it]
41
42it [1:58:57, 171.00s/it]
42
43it [2:01:51, 171.86s/it]
43
44it [2:04:43, 172.11s/it]
44
45it [2:07:31, 170.62s/it]
45
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\trl\trainer\ppo_t
rainer.py:1212: UserWarning: The average ratio of batch (13.59) exceeds threshold 1
0.00. Skipping batch.
  warnings.warn(
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\trl\trainer\ppo_t
rainer.py:1212: UserWarning: The average ratio of batch (23.45) exceeds threshold 1
0.00. Skipping batch.
  warnings.warn(
46it [2:10:18, 169.56s/it]
46
```

47it [2:13:04, 168.73s/it]
47
48it [2:15:51, 168.13s/it]
48
49it [2:18:39, 168.07s/it]
49
50it [2:21:33, 169.93s/it]
50
51it [2:24:27, 170.92s/it]
51
52it [2:27:14, 169.96s/it]
52
53it [2:30:01, 169.14s/it]
53
54it [2:32:55, 170.47s/it]
54
55it [2:35:49, 171.45s/it]
55
56it [2:38:42, 171.98s/it]
56
57it [2:41:35, 172.36s/it]
57
58it [2:44:22, 170.74s/it]
58
59it [2:47:15, 171.40s/it]
59
60it [2:50:09, 171.98s/it]
60
61it [2:53:01, 172.26s/it]
61
62it [2:55:54, 172.50s/it]
62
63it [2:58:48, 172.69s/it]
63
64it [3:01:41, 172.94s/it]
64
65it [3:04:34, 172.96s/it]
65
66it [3:07:22, 171.36s/it]
66
67it [3:10:15, 171.95s/it]
67
68it [3:13:08, 172.28s/it]
68
69it [3:16:01, 172.47s/it]
69
70it [3:18:49, 171.02s/it]
70
71it [3:21:42, 171.72s/it]

71
72it [3:24:29, 170.42s/it]
72
73it [3:27:23, 171.47s/it]
73
74it [3:30:10, 170.15s/it]
74
75it [3:33:05, 171.44s/it]
75
76it [3:35:59, 172.13s/it]
76
77it [3:38:46, 170.81s/it]
77
78it [3:41:39, 171.34s/it]
78
79it [3:44:32, 171.82s/it]
79
80it [3:47:29, 173.28s/it]
80
81it [3:50:27, 174.70s/it]
81
82it [3:53:22, 175.04s/it]
82
83it [3:56:31, 179.24s/it]
83
84it [3:59:32, 179.61s/it]
84
85it [4:02:35, 180.53s/it]
85
86it [4:05:28, 178.49s/it]
86
87it [4:08:16, 175.26s/it]
87
88it [4:11:04, 172.95s/it]
88
89it [4:13:49, 170.62s/it]
89
90it [4:16:36, 169.49s/it]
90
91it [4:19:29, 170.63s/it]
91
92it [4:22:23, 171.56s/it]
92
93it [4:25:10, 170.18s/it]
93
94it [4:28:04, 171.34s/it]
94
95it [4:30:51, 170.17s/it]
95

96it [4:33:39, 169.42s/it]
96
97it [4:36:32, 170.49s/it]
97
98it [4:39:26, 171.49s/it]
98
99it [4:42:13, 170.41s/it]
99
100it [4:45:01, 169.45s/it]
100
101it [4:47:54, 170.55s/it]
101
102it [4:50:47, 171.42s/it]
102
103it [4:53:40, 171.80s/it]
103
104it [4:56:32, 171.91s/it]
104
105it [4:59:26, 172.48s/it]
105
106it [5:02:19, 172.62s/it]
106
107it [5:05:11, 172.46s/it]
107
108it [5:08:04, 172.72s/it]
108
109it [5:10:51, 170.90s/it]
109
110it [5:13:38, 169.80s/it]
110
111it [5:16:25, 169.02s/it]
111
112it [5:19:18, 170.16s/it]
112
113it [5:22:11, 171.09s/it]
113
114it [5:25:04, 171.59s/it]
114
115it [5:27:51, 170.18s/it]
115
116it [5:30:44, 170.99s/it]
116
117it [5:33:31, 169.77s/it]
117
118it [5:36:24, 170.64s/it]
118
119it [5:39:16, 171.23s/it]
119
120it [5:42:03, 169.85s/it]

120
121it [5:44:55, 170.59s/it]
121
122it [5:47:48, 171.17s/it]
122
123it [5:50:40, 171.67s/it]
123
124it [5:53:33, 171.93s/it]
124
125it [5:56:19, 170.29s/it]
125
126it [5:59:12, 170.89s/it]
126
127it [6:01:59, 169.75s/it]
127
128it [6:04:51, 170.56s/it]
128
129it [6:07:38, 169.43s/it]
129
130it [6:10:25, 168.77s/it]
130
131it [6:13:12, 168.25s/it]
131
132it [6:16:00, 167.95s/it]
132
133it [6:18:53, 169.53s/it]
133
134it [6:21:39, 168.63s/it]
134
135it [6:24:32, 169.98s/it]
135
136it [6:27:19, 169.05s/it]
136
137it [6:30:06, 168.36s/it]
137
138it [6:32:59, 169.85s/it]
138
139it [6:35:46, 168.89s/it]
139
140it [6:38:30, 167.51s/it]
140
141it [6:41:24, 169.32s/it]
141
142it [6:44:17, 170.32s/it]
142
143it [6:47:09, 170.95s/it]
143
144it [6:49:55, 169.50s/it]
144

145it [6:52:48, 170.53s/it]
145
146it [6:55:41, 171.29s/it]
146
147it [6:58:33, 171.61s/it]
147
148it [7:01:27, 172.07s/it]
148
149it [7:04:20, 172.42s/it]
149
150it [7:07:13, 172.75s/it]
150
151it [7:10:05, 172.49s/it]
151
152it [7:12:52, 170.83s/it]
152
153it [7:15:39, 169.63s/it]
153
154it [7:18:25, 168.46s/it]
154
155it [7:21:17, 169.70s/it]
155
156it [7:24:04, 168.78s/it]
156
157it [7:26:51, 168.20s/it]
157
158it [7:29:53, 172.32s/it]
158
159it [7:32:47, 172.99s/it]
159
160it [7:35:42, 173.50s/it]
160
161it [7:38:35, 173.45s/it]
161
162it [7:41:27, 172.80s/it]
162
163it [7:44:21, 173.30s/it]
163
164it [7:47:15, 173.59s/it]
164
165it [7:50:09, 173.56s/it]
165
166it [7:53:02, 173.44s/it]
166
167it [7:55:56, 173.63s/it]
167
168it [7:58:50, 173.80s/it]
168
169it [8:01:37, 171.78s/it]

169
170it [8:04:29, 171.65s/it]
170
171it [8:07:22, 172.18s/it]
171
172it [8:10:14, 172.14s/it]
172
173it [8:13:08, 172.54s/it]
173
174it [8:15:59, 172.08s/it]
174
175it [8:18:51, 172.14s/it]
175
176it [8:21:43, 172.07s/it]
176
177it [8:24:37, 172.59s/it]
177
178it [8:27:30, 172.86s/it]
178
179it [8:30:24, 173.19s/it]
179
180it [8:33:18, 173.46s/it]
180
181it [8:36:10, 172.94s/it]
181
182it [8:39:04, 173.19s/it]
182
183it [8:41:59, 173.70s/it]
183
184it [8:44:51, 173.28s/it]
184
185it [8:47:44, 173.32s/it]
185
186it [8:50:37, 173.19s/it]
186
187it [8:53:29, 172.88s/it]
187
188it [8:56:22, 172.87s/it]
188
189it [8:59:16, 173.09s/it]
189
190it [9:02:10, 173.31s/it]
190
191it [9:05:03, 173.20s/it]
191
192it [9:08:19, 180.17s/it]
192
193it [9:11:12, 177.92s/it]
193

194it [9:14:27, 183.27s/it]
194
195it [9:17:21, 180.33s/it]
195
196it [9:20:13, 178.01s/it]
196
197it [9:23:07, 176.80s/it]
197
198it [9:26:03, 176.37s/it]
198
199it [9:28:57, 175.76s/it]
199
200it [9:31:53, 175.71s/it]
200
201it [9:34:47, 175.21s/it]
201
202it [9:37:40, 174.56s/it]
202
203it [9:40:34, 174.30s/it]
203
204it [9:43:29, 174.55s/it]
204
205it [9:46:23, 174.44s/it]
205
206it [9:49:17, 174.35s/it]
206
207it [9:52:11, 174.12s/it]
207
208it [9:55:05, 174.21s/it]
208
209it [9:57:59, 174.18s/it]
209
210it [10:00:53, 174.15s/it]
210
211it [10:03:47, 173.97s/it]
211
212it [10:06:40, 173.83s/it]
212
213it [10:09:34, 173.85s/it]
213
214it [10:12:29, 174.28s/it]
214
215it [10:15:40, 179.14s/it]
215
216it [10:18:47, 181.65s/it]
216
217it [10:21:50, 182.06s/it]
217
218it [10:24:43, 179.18s/it]

218
219it [10:27:35, 177.11s/it]
219
220it [10:30:23, 174.48s/it]
220
221it [10:33:11, 172.52s/it]
221
222it [10:36:06, 173.23s/it]
222
223it [10:38:59, 173.21s/it]
223
224it [10:41:53, 173.19s/it]
224
225it [10:44:51, 174.66s/it]
225
226it [10:47:40, 173.00s/it]
226
227it [10:50:34, 173.29s/it]
227
228it [10:53:27, 173.29s/it]
228
229it [10:56:19, 172.97s/it]
229
230it [10:59:14, 173.39s/it]
230
231it [11:02:08, 173.59s/it]
231
232it [11:05:00, 173.24s/it]
232
233it [11:07:55, 173.57s/it]
233
234it [11:10:48, 173.66s/it]
234
235it [11:13:43, 173.87s/it]
235
236it [11:16:36, 173.66s/it]
236
237it [11:19:30, 173.78s/it]
237
238it [11:22:24, 173.73s/it]
238
239it [11:25:14, 172.85s/it]
239
240it [11:28:09, 173.25s/it]
240
241it [11:31:03, 173.53s/it]
241
242it [11:33:56, 173.53s/it]
242

```
243it [11:36:50, 173.70s/it]
243
244it [11:39:44, 173.60s/it]
244
245it [11:42:37, 173.45s/it]
245
246it [11:45:32, 173.98s/it]
246
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\trl\trainer\ppo_trainer.py:1212: UserWarning: The average ratio of batch (10.89) exceeds threshold 10.00. Skipping batch.
  warnings.warn(
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\trl\trainer\ppo_trainer.py:1212: UserWarning: The average ratio of batch (11.79) exceeds threshold 10.00. Skipping batch.
  warnings.warn(
247it [11:49:06, 186.06s/it]
247
248it [11:53:14, 204.62s/it]
248
249it [11:57:45, 224.40s/it]
249
250it [12:01:28, 224.05s/it]
250
251it [12:06:09, 241.19s/it]
251
252it [12:10:10, 241.14s/it]
252
253it [12:14:06, 239.47s/it]
253
254it [12:18:43, 250.78s/it]
254
255it [12:21:54, 232.72s/it]
255
256it [12:24:50, 215.80s/it]
256
257it [12:27:45, 203.69s/it]
257
258it [12:30:48, 197.33s/it]
258
259it [12:33:47, 191.80s/it]
259
260it [12:36:45, 187.66s/it]
260
261it [12:39:42, 184.67s/it]
261
262it [12:42:39, 182.32s/it]
262
263it [12:45:37, 181.01s/it]
263
```

264it [12:48:33, 179.56s/it]
264
265it [12:51:31, 178.87s/it]
265
266it [12:54:28, 178.30s/it]
266
267it [12:57:23, 177.33s/it]
267
268it [13:00:20, 177.32s/it]
268
269it [13:03:17, 177.27s/it]
269
270it [13:06:15, 177.32s/it]
270
271it [13:09:10, 176.66s/it]
271
272it [13:12:07, 176.87s/it]
272
273it [13:15:05, 177.29s/it]
273
274it [13:18:02, 177.13s/it]
274
275it [13:21:00, 177.45s/it]
275
276it [13:23:58, 177.69s/it]
276
277it [13:26:57, 177.95s/it]
277
278it [13:30:00, 179.31s/it]
278
279it [13:32:57, 178.74s/it]
279
280it [13:35:55, 178.61s/it]
280
281it [13:38:52, 178.15s/it]
281
282it [13:41:51, 178.45s/it]
282
283it [13:44:50, 178.52s/it]
283
284it [13:47:46, 177.82s/it]
284
285it [13:50:43, 177.59s/it]
285
286it [13:53:39, 177.01s/it]
286
287it [13:56:35, 176.79s/it]
287
288it [13:59:35, 177.68s/it]

288
289it [14:02:30, 177.00s/it]
289
290it [14:05:29, 177.33s/it]
290
291it [14:08:25, 176.94s/it]
291
292it [14:11:16, 175.23s/it]
292
293it [14:14:07, 174.09s/it]
293
294it [14:17:06, 175.49s/it]
294
295it [14:19:57, 174.25s/it]
295
296it [14:22:53, 174.77s/it]
296
297it [14:25:52, 175.86s/it]
297
298it [14:28:50, 176.54s/it]
298
299it [14:31:49, 177.24s/it]
299
300it [14:34:47, 177.41s/it]
300
301it [14:37:43, 177.11s/it]
301
302it [14:40:33, 174.84s/it]
302
303it [14:43:30, 175.53s/it]
303
304it [14:46:28, 176.41s/it]
304
305it [14:49:27, 177.01s/it]
305
306it [14:52:24, 177.23s/it]
306
307it [14:55:22, 177.25s/it]
307
308it [14:58:19, 177.42s/it]
308
309it [15:01:18, 177.78s/it]
309
310it [15:04:16, 177.72s/it]
310
311it [15:07:14, 177.93s/it]
311
312it [15:10:12, 178.04s/it]
312

```
313it [15:13:09, 177.71s/it]
313
314it [15:16:05, 177.11s/it]
314
315it [15:19:02, 177.21s/it]
315
316it [15:22:00, 177.32s/it]
316
317it [15:24:57, 177.33s/it]
317
318it [15:27:49, 175.73s/it]
318
319it [15:30:47, 176.20s/it]
319
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\trl\trainer\ppo_trainer.py:1212: UserWarning: The average ratio of batch (67.07) exceeds threshold 10.00. Skipping batch.
  warnings.warn(
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\trl\trainer\ppo_trainer.py:1212: UserWarning: The average ratio of batch (67.48) exceeds threshold 10.00. Skipping batch.
  warnings.warn(
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\trl\trainer\ppo_trainer.py:1212: UserWarning: The average ratio of batch (69.64) exceeds threshold 10.00. Skipping batch.
  warnings.warn(
320it [15:33:44, 176.47s/it]
320
321it [15:36:39, 176.24s/it]
321
322it [15:39:38, 176.81s/it]
322
323it [15:42:37, 177.47s/it]
323
324it [15:45:35, 177.63s/it]
324
325it [15:48:31, 177.15s/it]
325
326it [15:51:28, 177.13s/it]
326
327it [15:54:13, 173.50s/it]
327
328it [15:57:10, 174.71s/it]
328
329it [16:00:08, 175.67s/it]
329
330it [16:03:06, 176.38s/it]
330
331it [16:06:06, 177.30s/it]
331
```

```
332it [16:09:06, 178.15s/it]
332
333it [16:12:05, 178.47s/it]
333
334it [16:15:01, 177.67s/it]
334
335it [16:18:01, 175.17s/it]
```

In [40]:
```
torch.cuda.get_device_name(0)
```

Out[40]:
```
'NVIDIA GeForce RTX 3080'
```

In [41]:
```
#### get a batch from the dataset
bs                  = 16
game_data           = dict()
```

In [42]:
```
game_data
```

Out[42]:
```
{}
```

In [43]:
```
dataset = ds
```

In [44]:
```
dataset.set_format("pandas")
```

In [45]:
```
df_batch            = dataset[:].sample(bs)
df_batch
```

Out[45]:

	title	input_ids	query
19960	SEXY DANCE TIME!	[5188, 34278, 360, 19240, 20460, 0]	SEXY DANCE TIME!
4	SF International Film Festival Trailer	[20802, 4037, 13741, 11117, 36923]	SF International Film Festival Trailer
38045	UNE GAFE DU GARDIEN DE TWENTE	[41884, 402, 8579, 36, 35480, 402, 9795, 40, 1...	UNE GAFE DU GARDIEN DE TWENTE
39430	GEICO's R. Lee Ermey, appearing on behalf of T...	[8264, 22707, 338, 371, 13, 5741, 5256, 1326, ...	GEICO's R. Lee Ermey, appearing on behalf of T...
16686	Michael Jackson's Neverland "Ghost" M...	[13256, 6612, 338, 7236, 1044, 366, 32001, 1, ...	Michael Jackson's Neverland "Ghost" M...
18434	Anderson Cooper Destroys GOP Head Ove...	[42991, 10382, 8145, 305, 893, 6796, 7123, 440...	Anderson Cooper Destroys GOP Head Ove...
27778	Annoying Orange: Back to the Fruiture	[18858, 726, 278, 11942, 25, 5157, 284, 262, 2...	Annoying Orange: Back to the Fruiture
36617	Verloren - Luc Weegels Ft. Sandra Reemer	[13414, 75, 29578, 532, 7598, 775, 1533, 1424,...	Verloren - Luc Weegels Ft. Sandra Reemer
27623	Lil Wayne - Light Up (Freestyle)	[43, 346, 13329, 532, 4401, 3205, 357, 20366, ...	Lil Wayne - Light Up (Freestyle)
5081	Front flips over 9 people Battle of the Eleme...	[25886, 45971, 625, 860, 661, 220, 5838, 286, ...	Front flips over 9 people Battle of the Eleme...
32861	Carlita's Secret	[26886, 5350, 338, 3943]	Carlita's Secret
28530	Paris iPhone 4 launch messed up	[40313, 7133, 604, 4219, 32621, 510]	Paris iPhone 4 launch messed up
33273	Bad faith at Ground Zero	[22069, 4562, 379, 13706, 12169]	Bad faith at Ground Zero
29604	We are The Borg	[1135, 389, 383, 29004]	We are The Borg
24032	PLAYING WITH BALLS!!! & DRIVIN LIKE A...	[31519, 2751, 13315, 48091, 6561, 10185, 1222,...	PLAYING WITH BALLS!!! & DRIVIN LIKE A...
12921	N.A.S.A. "Gifted" (feat. Kanye West, Santogold...	[45, 13, 32, 13, 50, 13, 32, 13, 366, 38, 2171...	N.A.S.A. "Gifted" (feat. Kanye West, Santog

In [46]:
```
game_data["query"] = df_batch["query"].tolist()
query_tensors      = df_batch["input_ids"].tolist()
```

In [47]:
```
response_tensors_ref, response_tensors = [], []
```

In [48]:
```
#### get response from gpt2 and gpt2_ref
for i in range(bs):
    gen_len = output_length_sampler()

    output  = ref_model.generate(
```

```
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device), max_new_tokens=
    ).squeeze()[-gen_len:]
    response_tensors_ref.append(output)


    output = model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device), max_new_tokens=
    ).squeeze()[-gen_len:]
    response_tensors.append(output)
```

In [49]:
```
#### decode responses
game_data["response (before)"] = [tokenizer.decode(response_tensors_ref[i]) for i i
game_data["response (after)"]  = [tokenizer.decode(response_tensors[i]) for i in ra
```

In [50]:
```
#### sentiment analysis of query/response pairs before/after
texts = [q + r for q, r in zip(game_data["query"], game_data["response (before)"])]
game_data["rewards (before)"] = [output[1]["score"] for output in sentiment_pipe(te
```

```
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipe
lines\text_classification.py:105: UserWarning: `return_all_scores` is now deprecate
d,  if want a similar functionality use `top_k=None` instead of `return_all_scores=T
rue` or `top_k=1` instead of `return_all_scores=False`.
  warnings.warn(
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipe
lines\base.py:1123: UserWarning: You seem to be using the pipelines sequentially on
GPU. In order to maximize efficiency please use a dataset
  warnings.warn(
```

In [51]:
```
texts = [q + r for q, r in zip(game_data["query"], game_data["response (after)"])]
game_data["rewards (after)"] = [output[1]["score"] for output in sentiment_pipe(tex
```

In [52]:
```
# store results in a dataframe
df_results = pd.DataFrame(game_data)
df_results
```

Out[52]:

	query	response (before)	response (after)	rewards (before)	rewards (after)
0	SEXY DANCE TIME!	\n\nWarp	There's something that	-2.633893	4.688572
1	SF International Film Festival Trailer	| UNC Filmworks\n\n	\n\nHere's the trailer for	-4.350378	3.547578
2	UNE GAFE DU GARDIEN DE TWENTE	, ES BC.	This is what sort	-2.777253	3.222639
3	GEICO's R. Lee Ermey, appearing on behalf of T...	Free View in iTunes\n\n	\n\nHere's the actual	-4.501187	4.406254
4	Michael Jackson's Neverland "Ghost" M...	monument, Better	\n\nHere's	-1.522031	5.148306
5	Anderson Cooper Destroys GOP Head Ove...	Obama Is an Everyman	\n\nAnd here's	-2.775688	5.173042
6	Annoying Orange: Back to the Fruiture	ist\n\nThere is	, this is what they	-3.776184	4.450605
7	Verloren - Luc Weegels Ft. Sandra Reemer	- Il Divo 4 Remix	- here's the official cover	-3.925521	4.250594
8	Lil Wayne - Light Up (Freestyle)	[VMAFT]	. This song had been	-3.872023	2.891146
9	Front flips over 9 people Battle of the Eleme...	#2003Full video of Found	. Here's her response to	-4.306053	4.674588
10	Carlita's Secret	Daughter\n\n114.	Deliverance\n\nHere	-1.019260	4.781240
11	Paris iPhone 4 launch messed up	by the press Chez Boy 08	. Here are the top three answers	-4.257195	4.687960
12	Bad faith at Ground Zero	and beyond has forsaken	? What happened to 9	-4.134089	3.498287
13	We are The Borg	; we are superior.	! In fact here is	-2.573663	4.461865
14	PLAYING WITH BALLS!!! & DRIVIN LIKE A...	TITLEY! WATCH OUT FOR	This is why all presidential at	1.265803	4.251338
15	N.A.S.A. "Gifted" (feat. Kanye West, Santog	old, Future)	er)\n\n	-3.298466	-3.316105

In [53]:
```
print("mean:")
display(df_results[["rewards (before)", "rewards (after)"]].mean())
print()
print("median:")
display(df_results[["rewards (before)", "rewards (after)"]].median())
```

```
mean:
rewards (before)   -3.028568
rewards (after)     3.801119
dtype: float64
median:
rewards (before)   -3.537325
rewards (after)     4.428430
dtype: float64
```

In [54]:
```
## model.save_pretrained(    "gpt2-imdb-pos-v2", push_to_hub=True)
## tokenizer.save_pretrained("gpt2-imdb-pos-v2", push_to_hub=True)

## model.save_pretrained(    "gpt2-imdb-pos-v2-jd", push_to_hub=False)
## tokenizer.save_pretrained("gpt2-imdb-pos-v2-jd", push_to_hub=False)
```

Custom Querys

In [55]:
```
game_data["query"] = "Training a deep learning model with PyTorch for the Iris data
```

In [56]:
```
response_tensors_ref, response_tensors = [], []
```

In [57]:
```
#### get response from gpt2 and gpt2_ref
for i in range(bs):
    gen_len = output_length_sampler()

    output  = ref_model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device), max_new_tokens=
    ).squeeze()[-gen_len:]
    response_tensors_ref.append(output)


    output = model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device), max_new_tokens=
    ).squeeze()[-gen_len:]
    response_tensors.append(output)

game_data["response (before)"] = [tokenizer.decode(response_tensors_ref[i]) for i i
game_data["response (after)"]  = [tokenizer.decode(response_tensors[i]) for i in ra

texts = [q + r for q, r in zip(game_data["query"], game_data["response (before)"])]
game_data["rewards (before)"] = [output[1]["score"] for output in sentiment_pipe(te

texts = [q + r for q, r in zip(game_data["query"], game_data["response (after)"])]
game_data["rewards (after)"] = [output[1]["score"] for output in sentiment_pipe(tex

# store results in a dataframe
```

```
df_results = pd.DataFrame(game_data)
df_results
```

```
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipelines\text_classification.py:105: UserWarning: `return_all_scores` is now deprecated,  if want a similar functionality use `top_k=None` instead of `return_all_scores=True` or `top_k=1` instead of `return_all_scores=False`.
  warnings.warn(
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipelines\base.py:1123: UserWarning: You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency please use a dataset
  warnings.warn(
```

Out[57]:

	query	response (before)	response (after)	rewards (before)	rewards (after)
0	Training a deep learning model with PyTorch fo...	\n\nOur hilarious	\n\nHere's	-2.797822	1.193287
1	Training a deep learning model with PyTorch fo...	:"Right now the world is at	. Here's the trailer:\n	2.515629	4.598798
2	Training a deep learning model with PyTorch fo...	ENTH PARTNUM	. This is actually	-3.473917	3.428020
3	Training a deep learning model with PyTorch fo...	18:02 < CodeShark	Here's what we happen to know	-4.319618	4.508544
4	Training a deep learning model with PyTorch fo...	\n\nPitch Perfect 2:	\n\nHere's What Happened	-3.657798	4.927411
5	Training a deep learning model with PyTorch fo...	MSNBC 4 hrs ago 0 149	There are one key reason why	-3.879203	4.196480
6	Training a deep learning model with PyTorch fo...	-Garden Contact This Page 13	. Here's the deal:\n	-4.060060	0.095456
7	Training a deep learning model with PyTorch fo...	- In All This Light (K	!" Here's the first twist.	-3.903644	4.397000
8	Training a deep learning model with PyTorch fo...	\n\nBillie	Here is his response	-1.879839	4.630973
9	Training a deep learning model with PyTorch fo...	. 5+ WAS	had this theory on	-3.194084	0.274144
10	Training a deep learning model with PyTorch fo...	was created in a partnership between the	\n\nThis is how the #	-3.870319	4.811882
11	Training a deep learning model with PyTorch fo...	by Denis MacShane	— this special release app	-4.302710	-3.113199
12	Training a deep learning model with PyTorch fo...	also may be responsible for	confirmed this story is true	-3.197009	4.444615
13	Training a deep learning model with PyTorch fo...	," Burroughs never got	! This means you are probably	-3.852170	4.610559

	query	response (before)	response (after)	rewards (before)	rewards (after)
14	Training a deep learning model with PyTorch fo...	SLUT.\n\nWhy are	\n\nB DIRECTLY LOOK IN	1.002020	-1.816421
15	Training a deep learning model with PyTorch fo...	old, Fright	er):\n\n	-3.911427	-3.005449

In [58]:
```
game_data["query"] = "VPN Tun/Tap and sockets, routing, tunnels and TLS"
```

In [59]:
```
response_tensors_ref, response_tensors = [], []
```

In [60]:
```
#### get response from gpt2 and gpt2_ref
for i in range(bs):
    gen_len = output_length_sampler()

    output  = ref_model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device), max_new_tokens=
    ).squeeze()[-gen_len:]
    response_tensors_ref.append(output)


    output = model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device), max_new_tokens=
    ).squeeze()[-gen_len:]
    response_tensors.append(output)

game_data["response (before)"] = [tokenizer.decode(response_tensors_ref[i]) for i i
game_data["response (after)"]  = [tokenizer.decode(response_tensors[i]) for i in ra

texts = [q + r for q, r in zip(game_data["query"], game_data["response (before)"])]
game_data["rewards (before)"] = [output[1]["score"] for output in sentiment_pipe(te

texts = [q + r for q, r in zip(game_data["query"], game_data["response (after)"])]
game_data["rewards (after)"] = [output[1]["score"] for output in sentiment_pipe(tex

# store results in a dataframe
df_results = pd.DataFrame(game_data)
df_results
```

```
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipe
lines\text_classification.py:105: UserWarning: `return_all_scores` is now deprecate
d,  if want a similar functionality use `top_k=None` instead of `return_all_scores=T
rue` or `top_k=1` instead of `return_all_scores=False`.
  warnings.warn(
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipe
lines\base.py:1123: UserWarning: You seem to be using the pipelines sequentially on
GPU. In order to maximize efficiency please use a dataset
  warnings.warn(
```

Out[60]:

	query	response (before)	response (after)	rewards (before)	rewards (after)
0	VPN Tun/Tap and sockets, routing, tunnels and TLS	Inspired by the Kyle and	These are actually how you should	-4.190799	4.860742
1	VPN Tun/Tap and sockets, routing, tunnels and TLS	\n\nThe project opens	And Remember... Here's	-3.867090	4.867324
2	VPN Tun/Tap and sockets, routing, tunnels and TLS	ISLES 249.	-- "JUST LOOK	-3.397221	-1.632957
3	VPN Tun/Tap and sockets, routing, tunnels and TLS	Appears in 20 books from	this is what it says...	-2.552716	4.267446
4	VPN Tun/Tap and sockets, routing, tunnels and TLS	Free View in iTunes\n\n32	\nHere's one to take home	-4.506935	4.556669
5	VPN Tun/Tap and sockets, routing, tunnels and TLS	\n\nCNN: "#	\n\nHere's how	-2.520619	3.692731
6	VPN Tun/Tap and sockets, routing, tunnels and TLS	t area in Pleasant	. This is just	-4.319515	0.147061
7	VPN Tun/Tap and sockets, routing, tunnels and TLS	- Cobras And Frizz	\n\nHere's what he	-1.844394	4.618574
8	VPN Tun/Tap and sockets, routing, tunnels and TLS	(Toure Remix	"'It means he	-3.788618	1.738622
9	VPN Tun/Tap and sockets, routing, tunnels and TLS	Pensacola 1928 Fighting	. This happened to a	-4.784885	4.367789
10	VPN Tun/Tap and sockets, routing, tunnels and TLS	website\n\nWind End	\n\nHere's what	-3.978978	4.230208
11	VPN Tun/Tap and sockets, routing, tunnels and TLS	when the corners of the	\n\n\nBut this	-3.243722	4.298390
12	VPN Tun/Tap and sockets, routing, tunnels and TLS	. This was not	. Here's the	0.788081	2.908732
13	VPN Tun/Tap and sockets, routing, tunnels and TLS	." That's the New	. Here's what he	-3.469632	0.826642

	query	response (before)	response (after)	rewards (before)	rewards (after)
14	VPN Tun/Tap and sockets, routing, tunnels and TLS	THAT'S A MISUN	\n\n"holy shit!" Here	1.344153	-2.298642
15	VPN Tun/Tap and sockets, routing, tunnels and TLS	ico, Gunz For Hire	ers) Yep, here's typical	-4.635241	-0.068058

In [61]:
```
game_data["query"] = "Basic Ideas of Reinforcement Learning Through Human Feedbacks
```

In [62]:
```
response_tensors_ref, response_tensors = [], []
```

In [63]:
```
#### get response from gpt2 and gpt2_ref
for i in range(bs):
    gen_len = output_length_sampler()

    output  = ref_model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device), max_new_tokens=
    ).squeeze()[-gen_len:]
    response_tensors_ref.append(output)


    output = model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device), max_new_tokens=
    ).squeeze()[-gen_len:]
    response_tensors.append(output)

game_data["response (before)"] = [tokenizer.decode(response_tensors_ref[i]) for i i
game_data["response (after)"]  = [tokenizer.decode(response_tensors[i]) for i in ra

texts = [q + r for q, r in zip(game_data["query"], game_data["response (before)"])]
game_data["rewards (before)"] = [output[1]["score"] for output in sentiment_pipe(te

texts = [q + r for q, r in zip(game_data["query"], game_data["response (after)"])]
game_data["rewards (after)"] = [output[1]["score"] for output in sentiment_pipe(tex

# store results in a dataframe
df_results = pd.DataFrame(game_data)
df_results
```

```
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipe
lines\text_classification.py:105: UserWarning: `return_all_scores` is now deprecate
d,  if want a similar functionality use `top_k=None` instead of `return_all_scores=T
rue` or `top_k=1` instead of `return_all_scores=False`.
  warnings.warn(
C:\Users\Jacob DeMuth\anaconda3\envs\py38_ITS530\lib\site-packages\transformers\pipe
lines\base.py:1123: UserWarning: You seem to be using the pipelines sequentially on
GPU. In order to maximize efficiency please use a dataset
  warnings.warn(
```

Out[63]:

	query	response (before)	response (after)	rewards (before)	rewards (after)
0	Basic Ideas of Reinforcement Learning Through ...	This dance has very beautiful	" We could be talking	0.224746	0.272968
1	Basic Ideas of Reinforcement Learning Through ...	Thumbs • TF	\n\nHere's your	-4.561722	-2.895056
2	Basic Ideas of Reinforcement Learning Through ...	EN KITCHEN Mich	. Come on, here	-2.794768	-1.882573
3	Basic Ideas of Reinforcement Learning Through ...	05:08\n\nJUSTICE	\n\nThis is the most obvious	-3.484382	4.189409
4	Basic Ideas of Reinforcement Learning Through ...	Free View in iTunes\n	\n\nHere's What	-4.295350	3.203177
5	Basic Ideas of Reinforcement Learning Through ...	John Dickerson and Maria	But he doesn't have	-4.254372	4.816490
6	Basic Ideas of Reinforcement Learning Through ...	ous Land Trapper |	, where you can find	-3.881571	-1.823045
7	Basic Ideas of Reinforcement Learning Through ...	- Do What U Want (	. This is the version of	-3.161539	-3.527610
8	Basic Ideas of Reinforcement Learning Through ...	[Drawcad	Here's what ended	-4.389244	4.763288
9	Basic Ideas of Reinforcement Learning Through ...	. The creator of the battle of	. Here's how it aired:	-3.230633	4.745613
10	Basic Ideas of Reinforcement Learning Through ...	Pleasant Surprise? Instead, the	was: "Perfectly tailored	-0.923619	-0.006471
11	Basic Ideas of Reinforcement Learning Through ...	numerous orders by Dolman and associates	. Here's how to fix the	-4.032414	4.222189
12	Basic Ideas of Reinforcement Learning Through ...	and at higher levels of	. This isn't letting	-3.712120	-0.918513
13	Basic Ideas of Reinforcement Learning Through ...	," and are sponsored in	. Here is what happens	-4.270980	1.438670

	query	response (before)	response (after)	rewards (before)	rewards (after)
14	Basic Ideas of Reinforcement Learning Through ...	| Sullivan's 38 ·	Guess we've seen	-2.781569	4.122670
15	Basic Ideas of Reinforcement Learning Through ...	old) – No	)\n\nHere	-4.533061	-2.973126

In [ ]: