Install dependencies

```
#@title Install dependencies
!pip install transformers
!pip install datasets
!pip install huggingface-hub
!pip install torch
!pip install wandb
!pip install lyricsgenius
!pip install aiohttp
!pip install langdetect
!pip install accelerate
!pip install --upgrade jax jaxlib
!pip install --upgrade git+https://github.com/google/flax.git
!pip install tqdm --upgrade
!pip install hf-lfs
!pip install ipywidgets
!pip install tensorboard
```

```
from datasets import load_dataset, Dataset, DatasetDict
from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer,TrainingArguments
from transformers import Trainer, TrainingArguments
import pathlib
import numpy as np
import random
```

```
def post_process(output_sequences):
    predictions = []
    generated_sequences = []

    max_repeat = 2

    # decode prediction
    for generated_sequence_idx, generated_sequence in enumerate(output_sequences):
        generated_sequence = generated_sequence.tolist()
        text = tokenizer.decode(generated_sequence, clean_up_tokenization_spaces=True, skip_special_tokens=True)
        generated_sequences.append(text.strip())

    for i, g in enumerate(generated_sequences):
        res = str(g).replace('\n\n\n', '\n').replace('\n\n', '\n')
        lines = res.split('\n')
        # print(lines)
        i = max_repeat
        while i != len(lines):
          remove_count = 0
          for index in range(0, max_repeat):
            # print(i - index - 1, i - index)
            if lines[i - index - 1] == lines[i - index]:
              remove_count += 1
          if remove_count == max_repeat:
            lines.pop(i)
            i -= 1
          else:
            i += 1
        predictions.append('\n'.join(lines))

    return predictions

def group_texts(examples):
    # Concatenate all texts.
    concatenated_examples = {k: sum(examples[k], []) for k in examples.keys()}
    total_length = len(concatenated_examples[list(examples.keys())[0]])
    # We drop the small remainder, we could add padding if the model supported it instead of this drop, you can
        # customize this part to your needs.
    total_length = (total_length // block_size) * block_size
    # Split by chunks of max_len.
    result = {
        k: [t[i : i + block_size] for i in range(0, total_length, block_size)]
        for k, t in concatenated_examples.items()
    }
    result["labels"] = result["input_ids"].copy()
    return result

def tokenize_function(examples):
    return tokenizer(examples["text"])
```

```
artist_name = "noize-mc"
check_dataset = True
num_train_epochs =  15

model_name = "noize-mc"


datasets = load_dataset("huggingartists/noize-mc")
train_percentage = 0.9
validation_percentage = 0.07
test_percentage = 0.03
train, validataion, test = np.split(datasets['train']['text'], [int(len(datasets['train']['text'])*train_percentage), int(le
datasets = DatasetDict(
    {
        'train': Dataset.from_dict({'text': list(train)}),
        'validataion': Dataset.from_dict({'text': list(validataion)}),
        'test': Dataset.from_dict({'text':list(test)})
    }
)
```

```
Downloading builder script: 100%|4.08k/4.08k [00:00<00:00, 226kB/s]
Downloading readme: 100%|7.17k/7.17k [00:00<00:00, 441kB/s]
Downloading data: 1.36M/1.36M [00:01<00:00,
```

```
tokenizer = AutoTokenizer.from_pretrained("gpt2")
model = AutoModelForCausalLM.from_pretrained("gpt2",cache_dir=pathlib.Path('cache').resolve())

tokenized_datasets = datasets.map(tokenize_function, batched=True, num_proc=1, remove_columns=["text"])

block_size = int(tokenizer.model_max_length / 4)

lm_datasets = tokenized_datasets.map(
    group_texts,
    batched=True,
    batch_size=1000,
    num_proc=1,
)
```

```
config.json: 100%|665/665 [00:00<00:00, 13.1kB/s]
vocab.json: 100%|1.04M/1.04M [00:00<00:00, 4.38MB/s]
merges.txt: 100%|456k/456k [00:00<00:00, 4.62MB/s]
tokenizer.json: 100%|1.36M/1.36M [00:00<00:00, 16.0MB/s]
config.json: 100%|665/665 [00:00<00:00, 27.5kB/s]
model.safetensors: 100%|548M/548M [00:02<00:00, 207MB/s]
generation_config.json: 100%|124/124 [00:00<00:00, 3.33kB/s]
Map: 100%|314/314 [00:03<00:00, 91.75 examples/s]
Token indices sequence length is longer than the specified maximum sequence length fo
Map: 100%|24/24 [00:00<00:00, 139.79 examples/s]
Map: 100%|11/11 [00:00<00:00, 74.19 examples/s]
```

```
seed_data = random.randint(0,2**32-1)
training_args = TrainingArguments(
    f"output/{model_name}",
    overwrite_output_dir=True,
    learning_rate=1.372e-4,
    weight_decay=0.01,
    num_train_epochs=num_train_epochs,
    save_total_limit=1,
    save_strategy='epoch',
    save_steps=1,
    seed=seed_data,
    logging_steps=5,
    report_to="tensorboard",
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=lm_datasets["train"],
)
```

```
!nvidia-smi
```

```
Wed Dec 20 18:03:34 2023
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   39C    P0              25W /  70W |    645MiB / 15360MiB |     16%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=========================================================================================|
+-----------------------------------------------------------------------------------------+
```

```
trainer.train()
```

Generation

```
start = "Вселенная"

num_sequences =  2
min_length =  100
max_length =   160
temperature = 0.3
top_p = 0.95
top_k = 50
repetition_penalty = 1.0

encoded_prompt = tokenizer(start, add_special_tokens=False, return_tensors="pt").input_ids
encoded_prompt = encoded_prompt.to(trainer.model.device)
# prediction
output_sequences = trainer.model.generate(
                        input_ids=encoded_prompt,
                        max_length=max_length,
                        min_length=min_length,
                        temperature=float(temperature),
                        top_p=float(top_p),
                        top_k=int(top_k),
                        do_sample=True,
                        repetition_penalty=repetition_penalty,
                        num_return_sequences=num_sequences
                        )
# Post-processing
predictions = post_process(output_sequences)

for num,prediction in enumerate(predictions):
  print(f"Result {num+1}")
  print(*prediction)
```

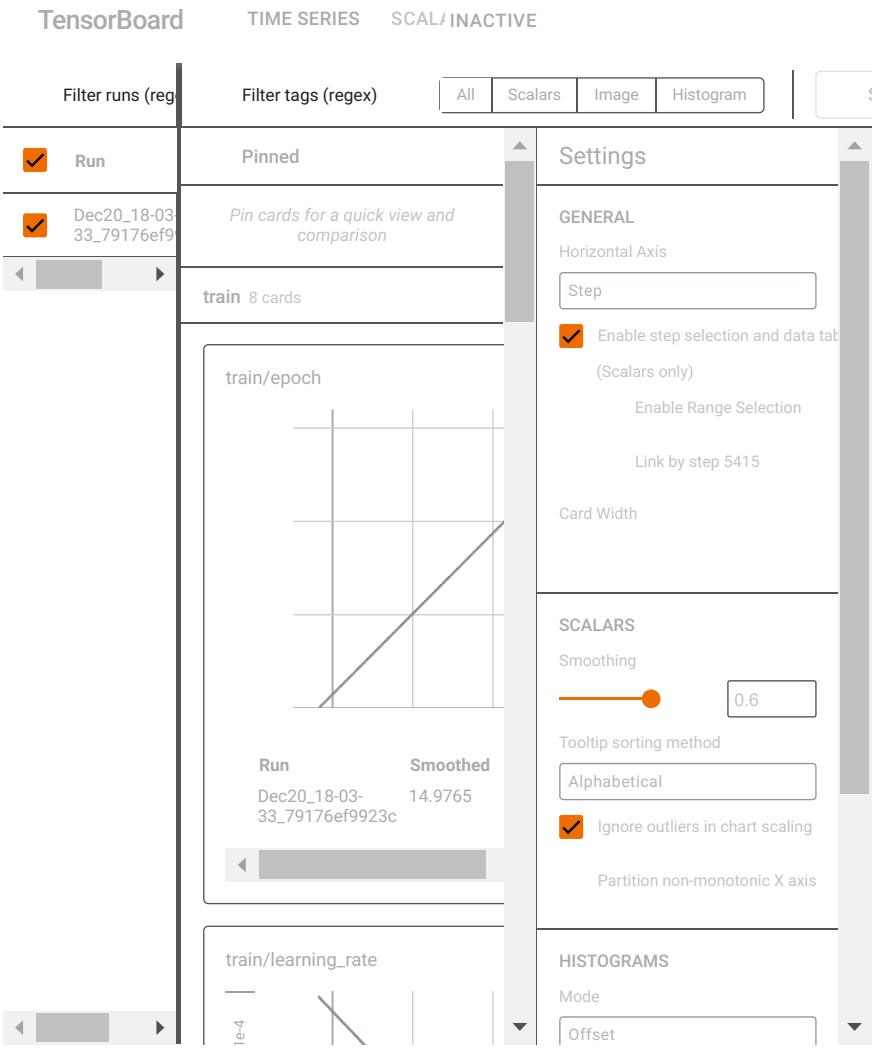
```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
```

```
Result 1
В с е л е н н а я  п о л н а я ,  п о л н а я  н а  к о л
 К а к  в о т  о н а ,  в с е г о  л и ш ь  т р е б я ,  к а к  в о д а
 О н а  с к а з а т ь  о  п о д о б н о й ,  п о п р о б у й  и  о б л а к а
 Н а  в е р ш и н е  м е с т а  х и т о в  к а р м а н а т о в  ◆
Result 2
В с е л е н н а я  п р о ш л а ,  ч т о  н а ш и  с т р а н н ы
 Н е  з а б ы в а й  п р и к и н е т ы  ш а г  в  т а к т  д а н н о й  и  г р у п п ы
 В  к а м о р к е ,  ч т о  з а  а л к а ш и  н а ш е г о  п р а в а
 Т р и  р а з а  в  н е д е л ю  р е
```

Tensorboard

```
%load_ext tensorboard
%tensorboard --logdir output/noize-mc/runs
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```




```
%reload_ext tensorboard
```