**Report**

In this report we experiment Hugging Face to generate code for NLP tasks using GPT-2.

**GPT-2 Architecture:**

def tokenize(element):

    outputs = tokenizer(

        element["content"],

        truncation=True,

        max\_length=context\_length,

        return\_overflowing\_tokens=True,

        return\_length=True,

    )

    input\_batch = []

    for length, input\_ids in zip(outputs["length"], outputs["input\_ids"]):

        if length == context\_length:

            input\_batch.append(input\_ids)

    return {"input\_ids": input\_batch}

tokenized\_datasets = raw\_datasets.map(

    tokenize, batched=True, remove\_columns=raw\_datasets["train"].column\_names

)

tokenized\_datasets

from transformers import AutoTokenizer, GPT2LMHeadModel, AutoConfig

config = AutoConfig.from\_pretrained(

    "gpt2",

    vocab\_sizse=len(tokenizer),

    n\_ctx=context\_length,

    bos\_token\_id=tokenizer.bos\_token\_id,

    eos\_token\_id=tokenizer.eos\_token\_id,

)

model = GPT2LMHeadModel(config)

model\_size = sum(t.numel() for t in model.parameters())

print(f"GPT-2 size: {model\_size/1000\*\*2:.1f}M parameters")

from transformers import DataCollatorForLanguageModeling

tokenizer.pad\_token = tokenizer.eos\_token

data\_collator = DataCollatorForLanguageModeling(tokenizer, mlm=False)

out = data\_collator([tokenized\_datasets["train"][i] for i in range(5)])

for key in out:

    print(f"{key} shape: {out[key].shape}")

**Then here we are with The First Experiment:**

With the first experiment we used these values for our variables (learning rate, optimizers, number of max steps).

args = TrainingArguments(

    output\_dir="codeparrot-ds",

    optim= 'adamw\_hf',

    per\_device\_train\_batch\_size=16,

    per\_device\_eval\_batch\_size=16,

    evaluation\_strategy="steps",

    eval\_steps=5\_000,

    logging\_steps=1,

    gradient\_accumulation\_steps=8,

    num\_train\_epochs=1,

    weight\_decay=0.1,

    warmup\_steps=100,

    lr\_scheduler\_type="cosine",

    learning\_rate=5e-4,

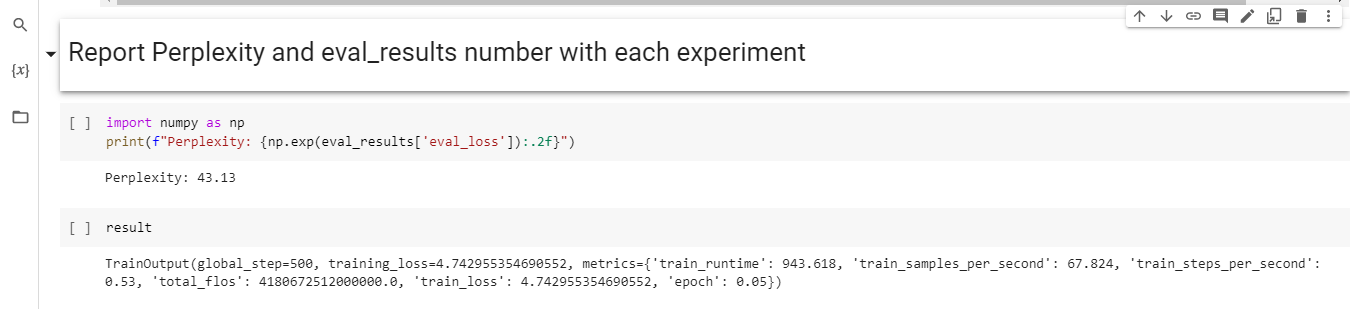
    save\_steps=100,

    fp16=True,

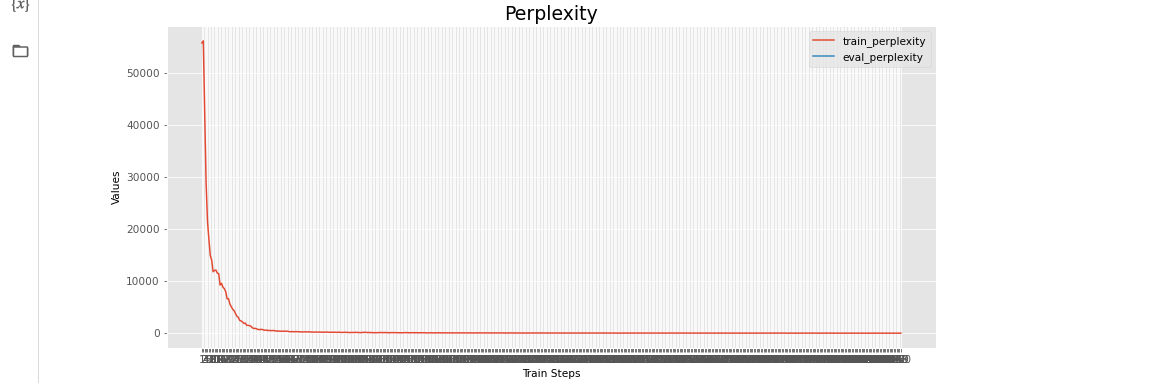
    max\_steps=500,

)

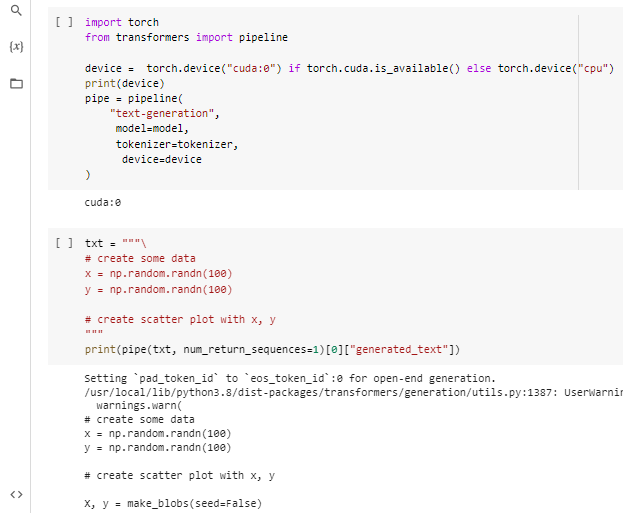
**The Perplexity and Evaluate Results:**

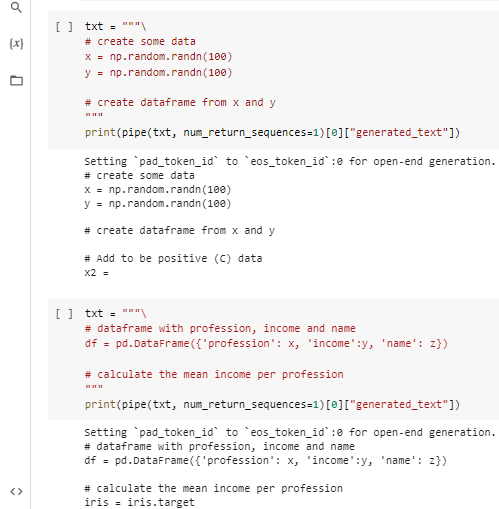


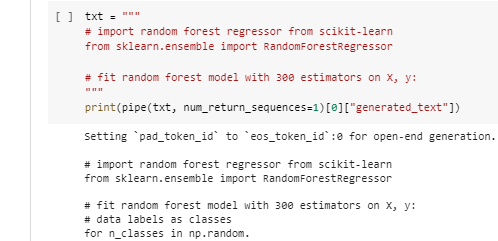
**Perplexity and Loss Graphs:**



**Code Prompts:**







**With The Second Experiment:**

We changed number of maximum steps to be 300 and number of epochs to be 2.

args = TrainingArguments(

    output\_dir="codeparrot-ds",

    optim= 'adamw\_hf',

    per\_device\_train\_batch\_size=16,

    per\_device\_eval\_batch\_size=16,

    evaluation\_strategy="steps",

    eval\_steps=5\_000,

    logging\_steps=1,

    gradient\_accumulation\_steps=8,

    num\_train\_epochs=2,

    weight\_decay=0.1,

    warmup\_steps=100,

    lr\_scheduler\_type="cosine",

    learning\_rate=5e-4,

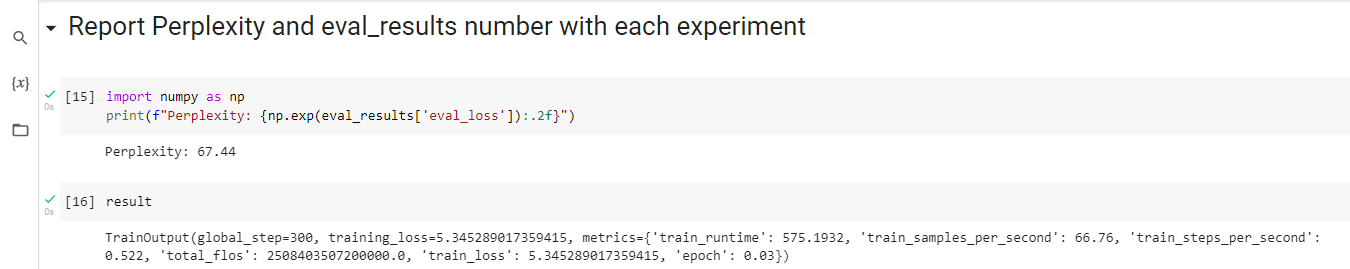
    save\_steps=100,

    fp16=True,

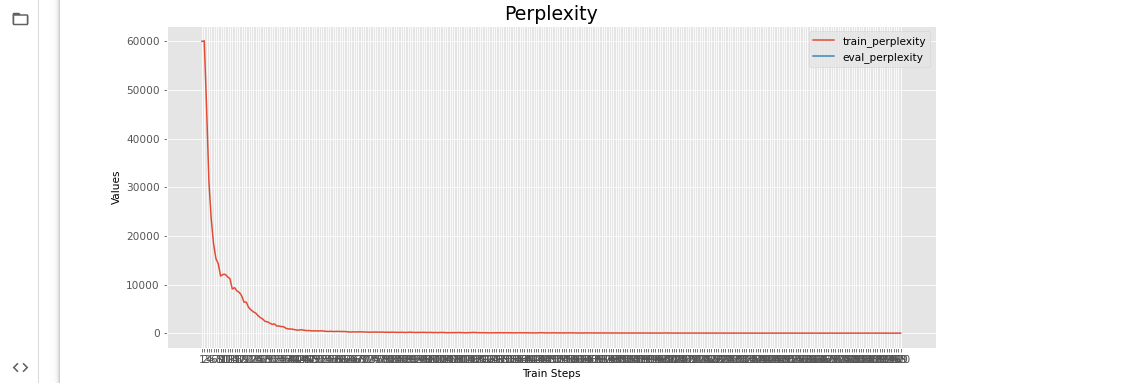
    max\_steps=300,

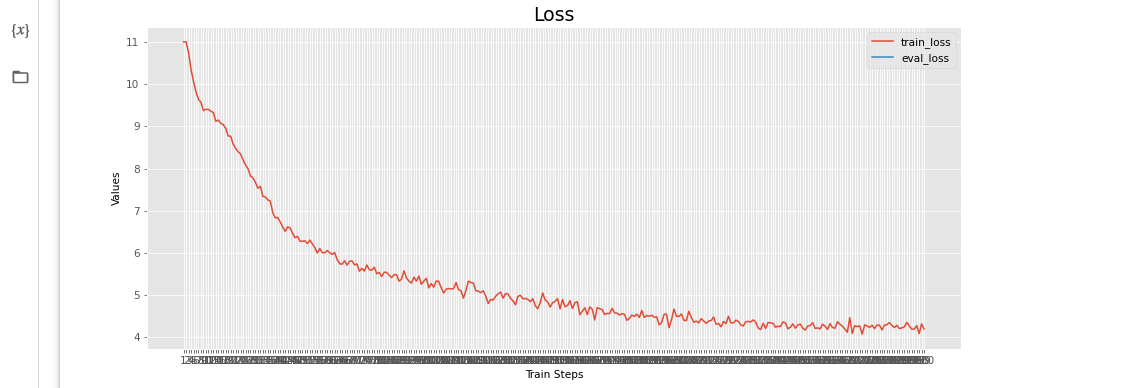
)

**The Perplexity and Evaluate Results:**

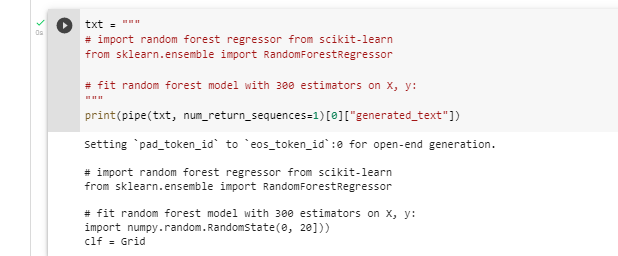
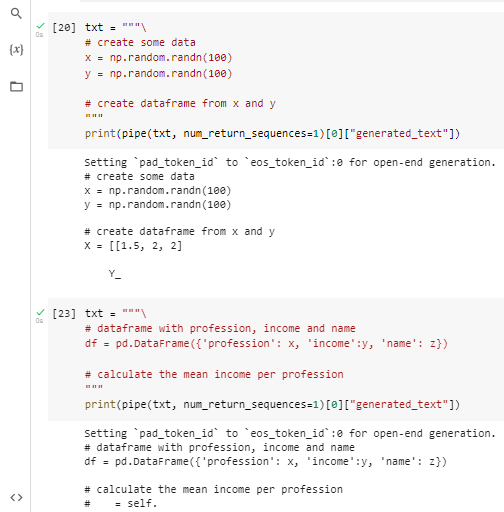
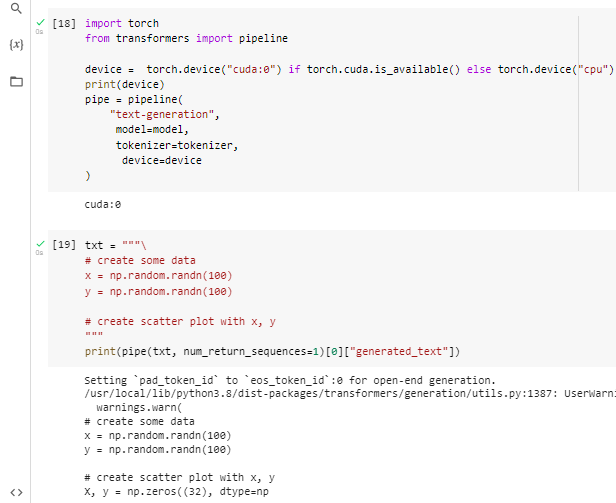


**Perplexity and Loss Graphs:**





**Code Prompts:**



**With The Third Experiment:**

We changed number of maximum steps to be 100.

args = TrainingArguments(

    output\_dir="codeparrot-ds",

    optim= 'adamw\_hf',

    per\_device\_train\_batch\_size=16,

    per\_device\_eval\_batch\_size=16,

    evaluation\_strategy="steps",

    eval\_steps=5\_000,

    logging\_steps=1,

    gradient\_accumulation\_steps=8,

    num\_train\_epochs=1,

    weight\_decay=0.1,

    warmup\_steps=100,

    lr\_scheduler\_type="cosine",

    learning\_rate=5e-4,

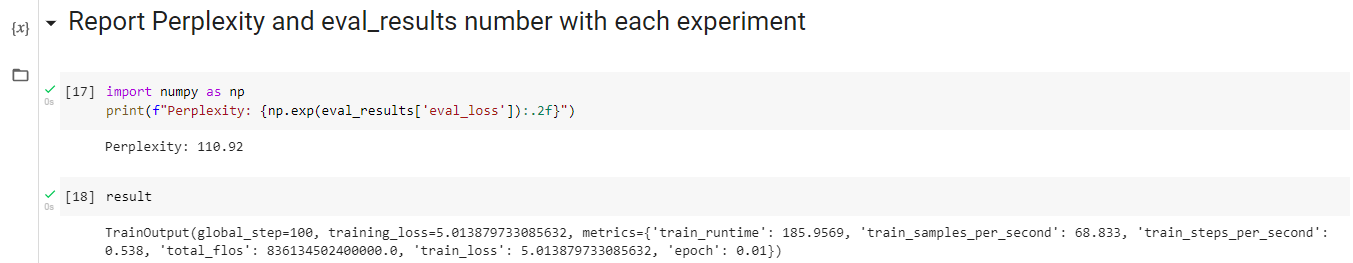
    save\_steps=100,

    fp16=True,

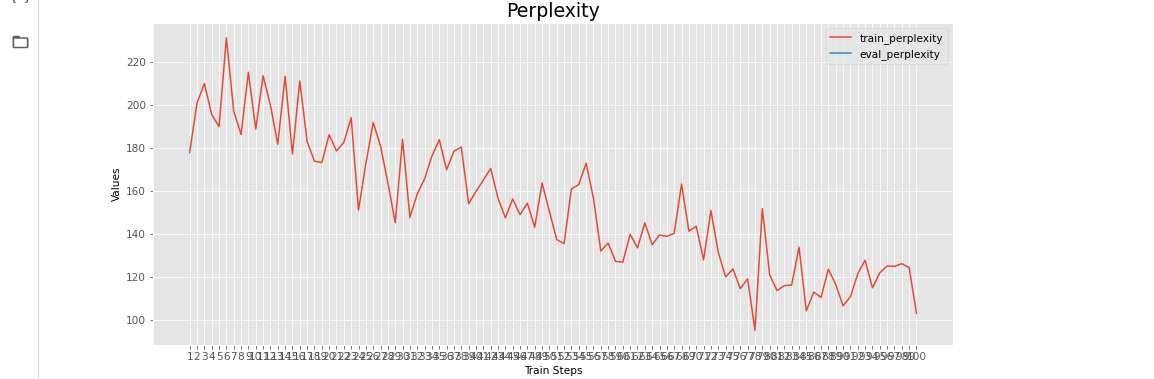
    max\_steps=100,

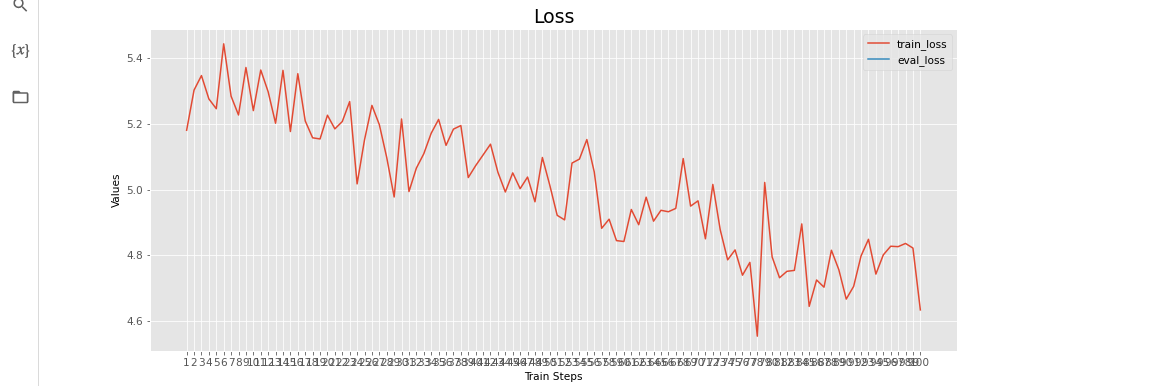
)

**The Perplexity and Evaluate Results:**

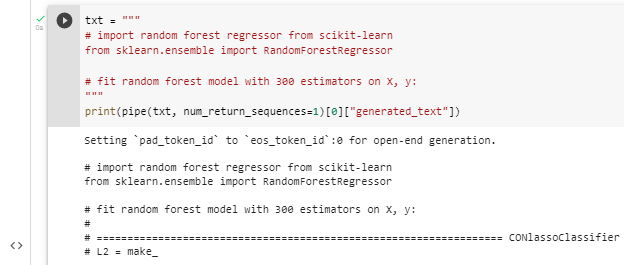
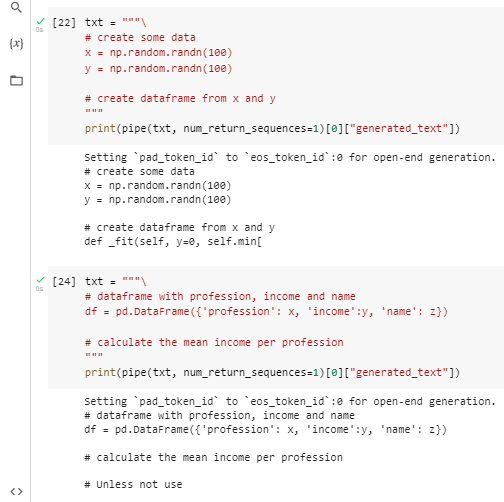
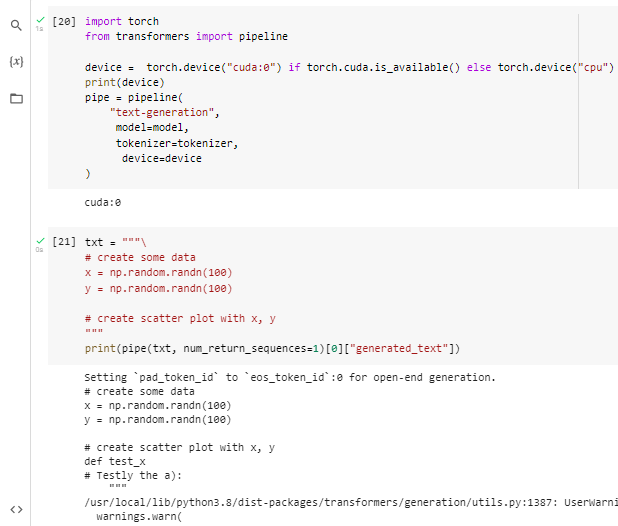


**Perplexity and Loss Graphs:**





**Code Prompts:**



**These are 3 experiments with different number of max steps of training to see the differences in perplexity and result values !**

**THANKS FOR READING THIS REPORT !**