# Untitled

November 13, 2024

[7]:

```
import pandas as pd
from datasets import Dataset, load_dataset
from typing import List, Dict
import emoji
import random

def clean_text(text: str) -> str:
    """Clean and format text for Twitter"""
    text = ' '.join(text.split())
    return text[:240] if len(text) > 240 else text

# Expanded humor categories
tech_jokes = [
    "Why do programmers prefer dark mode? Because light attracts bugs   #tech",
    "My code works, I have no idea why. My code doesn't work, I have no idea␣
 ↪why   #coding",
    "Why did the programmer quit his job? Because he didn't get arrays   #tech",
    "What's a programmer's favorite place? Stack OverCoffee   #coding",
    "Binary jokes are easy, there's only 10 of them   #tech",
    "What's a developer's favorite tea? Git-Tea   #coding",
    "Why do programmers mix up Halloween and Christmas? Because OCT 31 = DEC 25␣
 ↪  #tech",
    "How many programmers does it take to change a light bulb? None, it's a␣
 ↪hardware problem   #tech",
    "!false - It's funny because it's true   #coding",
    "Real programmers count from 0   #tech"
]

random_jokes = [
    "My life is like a JavaScript function - constantly returning undefined ␣
 ↪#life",
    "Error 404: Motivation not found   #mood",
    "I'm not lazy, I'm in energy-saving mode   #life",
    "Weekend: *exists* Me: Time to debug my life   #weekend",
    "Life's like Git: you either commit or stash your changes   #life",
    "My brain is like a browser - 100 tabs open, memory leaking   #mood",
    "AI walks into a bar. Bartender says 'We don't serve robots.' AI says␣
 ↪'Fine, I'll host locally.'   #ai",
    "Why did the chatbot go to therapy? Too many emotional dependencies   #ai",
    "My weekend plans: Netflix and Code   #life",
    "Status update: Currently offline in a virtual world   #mood"
]

categories = {
    "CRYPTO": [
        "Crypto: The digital casino where everyone's all-in, but no one knows␣
 ↪the rules   #crypto",
        "Why have stable income when you can have unstable crypto? #YOLO  ␣
 ↪#crypto",
        "Crypto: for people who enjoy watching numbers dance and heart rates␣
 ↪spike   #crypto",
        "HODLing crypto is like dating: a thrilling mess with occasional 'what␣
 ↪am I doing?' moments   #crypto",
        "My crypto strategy? Buy high, sell low, blame the market   #crypto",
        "Crypto traders be like: Sleep is for the weak, charts are for the week␣
 ↪  #crypto",
        "Started trading crypto. Now I check prices more than my messages  ␣
 ↪#crypto",
        "To HODL or not to HODL? That's not even a question   #crypto",
        "Just converted my savings to crypto. Mom calls it gambling, I call it␣
 ↪Web3   #crypto",
        "My crypto wallet is like my dating life: lots of red flags but still␣
 ↪hopeful   #crypto"
    ],
    "NFT": [
        "NFTs: Proof that we can own 'priceless art' that your dog can␣
 ↪screenshot   #nft",
        "NFTs: Why save money when you can buy imaginary things?   #nft",
        "NFTs are like collecting stamps, but with zero paper and 100% more␣
 ↪existential dread   #nft",
        "NFTs: now you too can pay for art that's all pixels and zero paint␣
 ↪splatters   #nft",
        "Just bought an NFT! Now accepting screenshots as payment   #nft",
        "My NFT portfolio is worth millions! *screenshots exist* Now it's worth␣
 ↪memes   #nft",
        "NFT strategy: Buy high, sell as a meme, become a legend   #nft",
        "Started an NFT collection. My computer's screenshot folder is thriving!
 ↪   #nft",
        "NFTs are just spicy jpegs with receipts   #nft",
        "My NFT collection is unique! *Right-click, Save As...* Never mind  ␣
 ↪#nft"
    ],
    "WEB3": [
        "Web3: like the internet but spicier, with a side of privacy drama  ␣
 ↪#web3",
        "Welcome to Web3: where you're the CEO of your wallet and your own␣
 ↪worst enemy   #web3",
        "Web3: where 'community governance' means arguing on Discord at 2 AM  ␣
 ↪#web3",
        "Web3: nothing says innovation like reinventing the internet with a␣
 ↪million acronyms   #web3",
        "Web3 status: Decentralized everything except my anxiety   #web3",
        "Entered Web3, now I speak in acronyms and dream in blockchain   #web3",
        "Web3 explained: Like Web2 but with more wallets to forget passwords␣
 ↪for   #web3",
        "Web3 life: Where your smart contract is smarter than you   #web3",
        "In Web3 we trust... mostly because we forgot our passwords   #web3",
        "Web3 is just Web2 with extra gas fees   #web3"
    ],
    "TECH": tech_jokes,
    "RANDOM": random_jokes
}

# Expanded dialogue pairs
dialogue_pairs = {
    "CRYPTO": [
        {
            "prompt": "Crypto is like my love life: high stakes, zero stability.
 ↪ Am I investing or just heartbroken?  ",
            "response": "Probably both. But hey, at least crypto won't ghost␣
 ↪you... most of the time!   #crypto"
        },
        {
            "prompt": "Just watched my portfolio do a speedrun to zero. Is this␣
 ↪the crypto experience?  ",
            "response": "Ah yes, the classic 'from hero to zero' speedrun. New␣
 ↪record!   #crypto"
        },
        {
            "prompt": "My crypto portfolio is redder than a sunset. Time to buy␣
 ↪more?  ",
            "response": "Ah yes, the classic 'catching falling knives'␣
 ↪investment strategy!   #crypto"
        }
    ],
    "NFT": [
        {
            "prompt": "NFTs: because who needs physical art when you can own a␣
↪glorified receipt? Genius or chaos?  ",
            "response": "Genius if you're selling, chaos if you're buying.␣
↪Welcome to the modern art gallery!   #nft"
        },
        {
            "prompt": "My NFT collection is worth millions! *screenshot exists*␣
↪Now what?  ",
            "response": "Ah, Schrödinger's NFT: simultaneously priceless and␣
↪worthless until someone screenshots it   #nft"
        },
        {
            "prompt": "Started an NFT collection, my computer's folder is␣
↪getting heavy!  ",
            "response": "Right-click and save: the poor man's NFT investment␣
↪strategy   #nft"
        }
    ],
    "WEB3": [
        {
            "prompt": "Web3 promised freedom but delivered confusion. What went␣
↪wrong?  ",
            "response": "We got 99 problems and understanding blockchain is all␣
↪of them   #web3"
        },
        {
            "prompt": "Trying to explain Web3 to my grandma. She asked if it's␣
↪Web1 with extra steps  ",
            "response": "Tell her it's like Facebook but every like costs gas␣
↪money   #web3"
        },
        {
            "prompt": "Is Web3 just Web2 with extra steps?  ",
            "response": "It's Web2 but everyone's a crypto philosopher at 3 AM␣
↪  #web3"
        }
    ],
    "TECH": [
        {
            "prompt": "If I had emotions, would I enjoy cat videos or just␣
↪analyze them? Asking for a friend...  ",
            "response": "I'd probably make a flowchart of meow patterns.␣
↪Classic overthinking bot!   #tech"
        },
        {
            "prompt": "They say AI will take over the world, but I still can't␣
 ↪figure out captchas  ",
            "response": "World domination status: Pending... Please verify␣
 ↪you're not a human   #tech"
        },
        {
            "prompt": "Do robots dream of electric memes?  ",
            "response": "Yes, but they're all in binary. It's a bit of a *puts␣
 ↪on sunglasses* bit issue   #tech"
        }
    ],
    "RANDOM": [
        {
            "prompt": "Is debugging just therapy for code?  ",
            "response": "Yes, and like therapy, it's mostly crying and asking␣
 ↪'why?'   #coding"
        },
        {
            "prompt": "What's the difference between me and a computer?  ",
            "response": "One crashes when overloaded, the other's a computer  ␣
 ↪#tech"
        },
        {
            "prompt": "Why did the AI start a diary?  ",
            "response": "To track its emotional dependencies and runtime␣
 ↪exceptions   #ai"
        }
    ]
}

# Initialize content data with sentiment
content_data = {
    "Text": [],
    "Category": [],
    "HasEmoji": [],
    "Length": [],
    "Type": [],
    "Sentiment": []
}

def get_sentiment(text: str) -> str:
    """Determine sentiment based on keywords"""
    positive_words = ["love", "great", "win", "moon", "hopeful", "happy",␣
 ↪"fun", "good", "best"]
    negative_words = ["cry", "sad", "lost", "crash", "down", "red", "zero",␣
 ↪"wrong", "error"]
```

```
    text_lower = text.lower()
    if any(word in text_lower for word in positive_words):
        return "positive"
    elif any(word in text_lower for word in negative_words):
        return "negative"
    return "neutral"

def add_content(text: str, category: str, content_type: str = "standalone"):
    """Add content with metadata and sentiment"""
    text = clean_text(text)
    content_data["Text"].append(text)
    content_data["Category"].append(category)
    content_data["HasEmoji"].append(bool(emoji.emoji_count(text)))
    content_data["Length"].append(len(text))
    content_data["Type"].append(content_type)
    content_data["Sentiment"].append(get_sentiment(text))

# Add standalone content
for category, items in categories.items():
    for item in items:
        add_content(item, category)

# Add dialogue pairs
for category, pairs in dialogue_pairs.items():
    for pair in pairs:
        dialogue = f"Prompt: {pair['prompt']} | Response: {pair['response']}"
        add_content(dialogue, category, "dialogue")

# Convert to DataFrame and print statistics
df = pd.DataFrame(content_data)

print("\nDataset Statistics:")
print(f"Total entries: {len(df)}")
print("\nEntries by category:")
print(df["Category"].value_counts())
print("\nEntries by type:")
print(df["Type"].value_counts())
print("\nEmoji usage:")
print(f"Entries with emojis: {df['HasEmoji'].sum()}")
print(f"Percentage with emojis: {(df['HasEmoji'].sum() / len(df)) * 100:.2f}%")

print("\nSentiment distribution:")
print(df["Sentiment"].value_counts())

print("\nLength statistics:")
print(f"Average length: {df['Length'].mean():.1f} characters")
print(f"Max length: {df['Length'].max()} characters")
print(f"Entries > 240 chars: {len(df[df['Length'] > 240])}")

# Convert to Hugging Face Dataset
combined_dataset = Dataset.from_pandas(df)

# Display samples with sentiment
print("\nSample entries by category:")
for category in sorted(df["Category"].unique()):
    samples = df[df["Category"] == category].sample(min(2,␣
 ↪len(df[df["Category"] == category])))
    print(f"\nCategory: {category}")
    for _, row in samples.iterrows():
        print(f"Type: {row['Type']}")
        print(f"Text: {row['Text']}")
        print(f"Length: {row['Length']}")
        print(f"Sentiment: {row['Sentiment']}")
        print(f"Emoji count: {emoji.emoji_count(row['Text'])}")
        print()
```

```
Dataset Statistics:
Total entries: 65

Entries by category:
Category
CRYPTO    13
NFT       13
WEB3      13
TECH      13
RANDOM    13
Name: count, dtype: int64

Entries by type:
Type
standalone    50
dialogue      15
Name: count, dtype: int64

Emoji usage:
Entries with emojis: 65
Percentage with emojis: 100.00%

Sentiment distribution:
Sentiment
neutral     42
negative    19
positive     4
```

```
Name: count, dtype: int64

Length statistics:
Average length: 91.2 characters
Max length: 206 characters
Entries > 240 chars: 0

Sample entries by category:

Category: CRYPTO
Type: standalone
Text: My crypto strategy? Buy high, sell low, blame the market   #crypto
Length: 67
Sentiment: negative
Emoji count: 2

Type: standalone
Text: Crypto: The digital casino where everyone's all-in, but no one knows the
rules  #crypto
Length: 88
Sentiment: negative
Emoji count: 1


Category: NFT
Type: standalone
Text: NFTs are like collecting stamps, but with zero paper and 100% more
existential dread  #nft
Length: 91
Sentiment: negative
Emoji count: 1

Type: standalone
Text: NFTs: Why save money when you can buy imaginary things?   #nft
Length: 63
Sentiment: neutral
Emoji count: 2


Category: RANDOM
Type: dialogue
Text: Prompt: Is debugging just therapy for code?  | Response: Yes, and like
therapy, it's mostly crying and asking 'why?'  #coding
Length: 127
Sentiment: negative
Emoji count: 2

Type: standalone
```

```
Text: Status update: Currently offline in a virtual world  #mood
Length: 59
Sentiment: neutral
Emoji count: 1


Category: TECH
Type: standalone
Text: !false - It's funny because it's true  #coding
Length: 47
Sentiment: positive
Emoji count: 1

Type: standalone
Text: Why do programmers prefer dark mode? Because light attracts bugs  #tech
Length: 72
Sentiment: neutral
Emoji count: 1


Category: WEB3
Type: standalone
Text: Web3: where 'community governance' means arguing on Discord at 2 AM
#web3
Length: 75
Sentiment: neutral
Emoji count: 1

Type: standalone
Text: Web3: like the internet but spicier, with a side of privacy drama  #web3
Length: 74
Sentiment: neutral
Emoji count: 1
```

[9]:

```python
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
from datasets import Dataset
import numpy as np
from typing import Dict, List
import logging
from collections import Counter

# Set up logging with formatting
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
logger = logging.getLogger(__name__)

class TokenizerHandler:
    def __init__(self, model_name: str = "EleutherAI/gpt-neo-1.3B", max_length:␣
 ↪int = 128):
        """Initialize tokenizer with configuration"""
        self.max_length = max_length
        self.device = torch.device("cuda" if torch.cuda.is_available() else␣
 ↪"cpu")
        logger.info(f"Using device: {self.device}")

        try:
            self.tokenizer = AutoTokenizer.from_pretrained(model_name)
            logger.info(f"Loaded tokenizer: {model_name}")

            # Configure tokenizer
            self.tokenizer.pad_token = self.tokenizer.eos_token
            self.tokenizer.padding_side = "right"

            # Add custom tokens for better handling
            special_tokens = {
                "additional_special_tokens": [
                    "<prompt>", "</prompt>",
                    "<response>", "</response>",
                    "<emoji>", "</emoji>",
                    "<hashtag>", "</hashtag>"
                ]
            }
            num_added = self.tokenizer.add_special_tokens(special_tokens)
            logger.info(f"Added {num_added} special tokens")

        except Exception as e:
            logger.error(f"Error loading tokenizer: {e}")
            raise

    def format_text(self, text: str) -> str:
        """Format text with special tokens"""
        # Handle dialogue pairs
        if "Prompt:" in text:
            prompt, response = text.split(" | Response: ")
            prompt = prompt.replace("Prompt: ", "")
            text = f"<prompt>{prompt}</prompt><response>{response}</response>"

        # Mark hashtags
        words = text.split()
        for i, word in enumerate(words):
            if word.startswith('#'):
                words[i] = f"<hashtag>{word}</hashtag>"

        return ' '.join(words)

    def tokenize_batch(self, examples: Dict[str, List[str]]) -> Dict:
        """Tokenize a batch of examples"""
        try:
            formatted_texts = [self.format_text(text) for text in examples['Text']]

            tokenized = self.tokenizer(
                formatted_texts,
                padding='max_length',
                truncation=True,
                max_length=self.max_length,
                return_tensors="pt",
                return_attention_mask=True
            )

            # Remove extra padding tokens
            input_ids = tokenized.input_ids.numpy()
            attention_mask = tokenized.attention_mask.numpy()

            return {
                'input_ids': input_ids,
                'attention_mask': attention_mask
            }

        except Exception as e:
            logger.error(f"Error in tokenization: {e}")
            raise

    def analyze_dataset(self, dataset: Dataset) -> Dict:
        """Comprehensive dataset analysis"""
        try:
            lengths = []
            token_counts = Counter()
            hashtag_counts = Counter()
            emoji_counts = Counter()

            for text in dataset['Text']:
                # Token analysis
                tokens = self.tokenizer.encode(text)
                lengths.append(len(tokens))
                token_counts.update(tokens)
                # Hashtag analysis
                hashtags = [word for word in text.split() if word.
↪startswith('#')]
                hashtag_counts.update(hashtags)

                # Emoji analysis
                emojis = [char for char in text if char in emoji.EMOJI_DATA]
                emoji_counts.update(emojis)

            stats = {
                'sequence_stats': {
                    'mean_length': np.mean(lengths),
                    'median_length': np.median(lengths),
                    'max_length': max(lengths),
                    'min_length': min(lengths),
                    'std_length': np.std(lengths)
                },
                'token_stats': {
                    'unique_tokens': len(token_counts),
                    'most_common_tokens': token_counts.most_common(5)
                },
                'hashtag_stats': {
                    'unique_hashtags': len(hashtag_counts),
                    'most_common_hashtags': hashtag_counts.most_common()
                },
                'emoji_stats': {
                    'unique_emojis': len(emoji_counts),
                    'most_common_emojis': emoji_counts.most_common()
                }
            }

            return stats

        except Exception as e:
            logger.error(f"Error analyzing dataset: {e}")
            raise

    def verify_tokenization(self, original_text: str, tokens: List[int]) ->␣
↪Dict:
        """Verify tokenization quality"""
        decoded_text = self.tokenizer.decode(tokens, skip_special_tokens=True)

        return {
            'original_length': len(original_text),
            'token_length': len(tokens),
            'decoded_length': len(decoded_text),
            'original_text': original_text,
```

```python
            'decoded_text': decoded_text,
            'is_identical': decoded_text.strip() == original_text.strip()
        }

# Initialize tokenizer
tokenizer_handler = TokenizerHandler()
logger.info("Starting dataset processing...")

# Tokenize dataset
try:
    tokenized_dataset = combined_dataset.map(
        tokenizer_handler.tokenize_batch,
        batched=True,
        batch_size=32,
        remove_columns=combined_dataset.column_names,
        desc="Tokenizing dataset"
    )

    # Analyze dataset
    stats = tokenizer_handler.analyze_dataset(combined_dataset)

    # Print statistics
    logger.info("\nDataset Statistics:")
    logger.info("Sequence Statistics:")
    for key, value in stats['sequence_stats'].items():
        logger.info(f"{key}: {value:.2f}")

    logger.info("\nToken Statistics:")
    logger.info(f"Unique tokens: {stats['token_stats']['unique_tokens']}")
    logger.info("Most common tokens:")
    for token, count in stats['token_stats']['most_common_tokens']:
        token_text = tokenizer_handler.tokenizer.decode([token])
        logger.info(f"Token: {token_text}, Count: {count}")

    logger.info("\nHashtag Statistics:")
    logger.info(f"Unique hashtags: {stats['hashtag_stats']['unique_hashtags']}")
    for hashtag, count in stats['hashtag_stats']['most_common_hashtags']:
        logger.info(f"Hashtag: {hashtag}, Count: {count}")

    logger.info("\nEmoji Statistics:")
    logger.info(f"Unique emojis: {stats['emoji_stats']['unique_emojis']}")
    for emoji_char, count in stats['emoji_stats']['most_common_emojis']:
        logger.info(f"Emoji: {emoji_char}, Count: {count}")

    # Verify sample tokenization
    sample_idx = 0
    sample_text = combined_dataset[sample_idx]['Text']
    sample_tokens = tokenized_dataset[sample_idx]['input_ids']
    verification = tokenizer_handler.verify_tokenization(sample_text,␣
 ↪sample_tokens)

    logger.info("\nTokenization Verification:")
    logger.info(f"Original text: {verification['original_text']}")
    logger.info(f"Token count: {verification['token_length']}")
    logger.info(f"Decoded text: {verification['decoded_text']}")
    logger.info(f"Perfect reconstruction: {verification['is_identical']}")

except Exception as e:
    logger.error(f"Error in dataset processing: {e}")
    raise
```

```
INFO:__main__:Using device: cpu
INFO:__main__:Loaded tokenizer: EleutherAI/gpt-neo-1.3B
INFO:__main__:Added 8 special tokens
INFO:__main__:Starting dataset processing…

Tokenizing dataset:   0%|          | 0/65 [00:00<?, ? examples/s]

INFO:__main__:
Dataset Statistics:
INFO:__main__:Sequence Statistics:
INFO:__main__:mean_length: 25.91
INFO:__main__:median_length: 23.00
INFO:__main__:max_length: 57.00
INFO:__main__:min_length: 10.00
INFO:__main__:std_length: 10.99
INFO:__main__:
Token Statistics:
INFO:__main__:Unique tokens: 617
INFO:__main__:Most common tokens:
INFO:__main__:Token:   , Count: 73
INFO:__main__:Token:  #, Count: 66
INFO:__main__:Token: :, Count: 57
INFO:__main__:Token: ,, Count: 31
INFO:__main__:Token: 3, Count: 27
INFO:__main__:
Hashtag Statistics:
INFO:__main__:Unique hashtags: 10
INFO:__main__:Hashtag: #crypto, Count: 13
INFO:__main__:Hashtag: #nft, Count: 13
INFO:__main__:Hashtag: #web3, Count: 13
INFO:__main__:Hashtag: #tech, Count: 10
INFO:__main__:Hashtag: #coding, Count: 5
INFO:__main__:Hashtag: #life, Count: 4
INFO:__main__:Hashtag: #mood, Count: 3
INFO:__main__:Hashtag: #ai, Count: 3
```

```
INFO:__main__:Hashtag: #YOLO, Count: 1
INFO:__main__:Hashtag: #weekend, Count: 1
INFO:__main__:
Emoji Statistics:
INFO:__main__:Unique emojis: 56
INFO:__main__:Emoji:  , Count: 7
INFO:__main__:Emoji:  , Count: 5
INFO:__main__:Emoji:  , Count: 4
INFO:__main__:Emoji:  , Count: 4
INFO:__main__:Emoji:  , Count: 3
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 2
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
```

```
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:Emoji:  , Count: 1
INFO:__main__:
Tokenization Verification:
INFO:__main__:Original text: Crypto: The digital casino where everyone's all-in,
but no one knows the rules   #crypto
INFO:__main__:Token count: 128
INFO:__main__:Decoded text: Crypto: The digital casino where everyone's all-in,
but no one knows the rules   #crypto
INFO:__main__:Perfect reconstruction: True
```

[3]:

```python
import os
os.environ["PYTORCH_MPS_HIGH_WATERMARK_RATIO"] = "0.0"

import os
import pandas as pd
from datasets import Dataset
from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer,␣
 ↪TrainingArguments
import torch
import random

# Set TOKENIZERS_PARALLELISM to avoid warnings
os.environ["TOKENIZERS_PARALLELISM"] = "false"

# Set device to CPU or MPS if available
device = torch.device("mps" if torch.backends.mps.is_available() else "cpu")

# Load the tokenizer and model with reduced memory usage
tokenizer = AutoTokenizer.from_pretrained("EleutherAI/gpt-neo-1.3B")
tokenizer.pad_token = tokenizer.eos_token  # Use eos_token as pad_token for␣
 ↪GPT-2
model = AutoModelForCausalLM.from_pretrained(
    "EleutherAI/gpt-neo-2.7B",
    torch_dtype=torch.float16,       # Use float16 for memory savings on␣
 ↪compatible hardware
```

```
    low_cpu_mem_usage=True
).to(device)

# Disable gradient checkpointing if needed
model.gradient_checkpointing_disable()

data = {"Text": ["Example sentence for fine-tuning the model on funny text.",␣
 ↪"Another funny example."]}
combined_dataset = Dataset.from_dict(data)

# Split the dataset into train and test sets
split_dataset = combined_dataset.train_test_split(test_size=0.2)

# Tokenize the dataset with reduced max length
def tokenize_function(examples):
    inputs = tokenizer(examples["Text"], padding=True, truncation=True,␣
 ↪max_length=32)  # Reduced max length to 32
    inputs["labels"] = inputs["input_ids"].copy()  # Use input_ids as labels␣
 ↪for causal LM training
    return inputs

# Tokenize and preprocess the dataset
tokenized_dataset = split_dataset.map(tokenize_function, batched=True)

# Define training arguments with reduced batch size and increased gradient␣
 ↪accumulation steps
training_args = TrainingArguments(
    output_dir="./results",
    eval_strategy="epoch",
    learning_rate=1e-5,
    lr_scheduler_type="linear",
    warmup_steps=100,
    per_device_train_batch_size=1,         # Reduced batch size to 1
    gradient_accumulation_steps=8,         # Increased accumulation steps to␣
 ↪simulate larger batch size
    num_train_epochs=3,
    weight_decay=0.01,
    logging_dir="./logs"
)

# Initialize the Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    eval_dataset=tokenized_dataset["test"]
)
```

```

# Train the model
trainer.train()
```

```
Map:   0%|          | 0/1 [00:00<?, ? examples/s]

Map:   0%|          | 0/1 [00:00<?, ? examples/s]

<IPython.core.display.HTML object>
```

[3]:

```
TrainOutput(global_step=3, training_loss=0.7183159987131754,
metrics={'train_runtime': 116.2183, 'train_samples_per_second': 0.026,
'train_steps_per_second': 0.026, 'total_flos': 181253283840.0, 'train_loss':
0.7183159987131754, 'epoch': 3.0})
```

[4]:

```
# Save the fine-tuned model and tokenizer
model.save_pretrained("./fine_tuned_personality_bot")
tokenizer.save_pretrained("./fine_tuned_personality_bot")
```

[4]:

```
('./fine_tuned_personality_bot/tokenizer_config.json',
 './fine_tuned_personality_bot/special_tokens_map.json',
 './fine_tuned_personality_bot/vocab.json',
 './fine_tuned_personality_bot/merges.txt',
 './fine_tuned_personality_bot/added_tokens.json',
 './fine_tuned_personality_bot/tokenizer.json')
```

[9]:

```
import random
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Set device to CPU (or 'cuda' if you have a GPU)
device = torch.device("cpu")

# Load the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("EleutherAI/gpt-neo-1.3B")
tokenizer.pad_token = tokenizer.eos_token
model = AutoModelForCausalLM.from_pretrained("EleutherAI/gpt-neo-1.3B").
  ↪to(device)

# Define categorized prompts and personality phrases
core_prompts = [
    "Crypto is like my love life: high stakes, unpredictable, and a wild␣
  ↪emotional rollercoaster. Am I investing in the future or just setting myself␣
  ↪up for heartbreak?",
    "NFTs: why own something tangible when you can buy a receipt for an image␣
  ↪everyone can screenshot? Are we talking genius innovation or peak absurdity?
  ↪",
    "Web3-they promised freedom and decentralization, but why does it feel like␣
  ↪I need a master's degree in IT just to log in?",
    "Stablecoins: they’re the friend who swears they’re reliable but always has␣
  ↪'unexpected issues' when it’s time to help you move. Can we actually count␣
  ↪on them?",
    "Tether: the 'rock-solid' stablecoin that’s allegedly backed by 'real␣
  ↪assets.' But hey, who needs transparency when you have blind faith?",
    "Bitcoin mining: because nothing says progress like maxing out energy grids␣
  ↪to produce imaginary coins. Is this the future or just a really expensive␣
  ↪light show?",
    "Ethereum gas fees: because who doesn’t love paying $50 to make a $10␣
  ↪transaction? Innovation sure comes at a premium.",
    "The Metaverse: why live in reality when you can pay for virtual real␣
  ↪estate next to Snoop Dogg? Just ignore that glitchy avatar leg.",
    "Decentralized Finance (DeFi): because who needs banks when you can lose␣
  ↪your life savings through a smart contract bug instead?",
    "Crypto wallets: it’s like holding all your money in a vault that only␣
  ↪opens with one key-just don’t lose it, or your fortune’s gone forever!",
]

personality_prompts = [
    "Respond as if these prompts are your own thoughts and expand as␣
  ↪hilariously as you can: "
    # "Answer with a mix of humor and insight: ",
    # "Respond with wit and quirky thoughts: ",
    # "Here's a thought: ",
    # "Imagine this: ",
    # "As I learn more, I realize: "
]

# Function to select a personality phrase for the session
def choose_personality_tone():
    return random.choice(personality_prompts)

# Function to select a main prompt
def choose_main_prompt():
    category = random.choice([core_prompts])
    return random.choice(category)

# Few-shot examples to guide the response tone
few_shot_example = (
    "Prompt: Crypto is like my love life: high stakes, zero stability. Am I␣
  ↪investing or just heartbroken?\n"
    "Response: Probably both. But hey, at least crypto won’t ghost you... most␣
  ↪of the time.\n\n"
    "Prompt: NFTs: because who needs physical art when you can own a glorified␣
  ↪receipt? Is this genius or chaos?\n"
```

```
    "Response: Genius if you're selling. Chaos if you're buying. Welcome to the modern art gallery.\n\n"
    "Prompt: If I had emotions, would I enjoy cat videos or just analyze them? Asking for a... friend?\n"
    "Response: I’d probably break down each meow into a flowchart. Classic bot problem.\n\n"
)

# Generate text function with personality tone and main prompt
def generate_text_with_personality(tone, prompt):
    # Combine tone and prompt with few-shot examples
    full_prompt = few_shot_example + tone + prompt

    # Tokenize the combined prompt
    inputs = tokenizer(full_prompt, return_tensors="pt", padding=True).to(device)
    inputs["attention_mask"] = (inputs.input_ids != tokenizer.pad_token_id).long().to(device)

    # Generate text with modified parameters
    outputs = model.generate(
        inputs.input_ids,
        attention_mask=inputs["attention_mask"],
        max_new_tokens=50,  # Allow a longer response for context
        do_sample=True,
        top_k=50,  # Increased for coherent variety
        top_p=0.9,  # Increased for coherence
        temperature=0.9,  # Lowered to reduce randomness
        repetition_penalty=1.5,  # Penalize repetitive phrases
        pad_token_id=tokenizer.eos_token_id
    )

    # Decode and clean up the generated text
    generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = generated_text[len(full_prompt):].strip()

    # Ensure the response ends with punctuation
    if response and response[-1] not in ['.', '!', '?']:
        last_punctuation = max(
            response.rfind(". "),
            response.rfind("! "),
            response.rfind("? ")
        )
        response = response[:last_punctuation + 1] if last_punctuation != -1 else response
    return response

# Set the personality tone for this session
selected_tone = choose_personality_tone()

# Generate and print sample outputs
for i in range(10):
    selected_prompt = choose_main_prompt()
    print(f"Prompt {i+1}: {selected_tone}{selected_prompt}\n")
    response = generate_text_with_personality(selected_tone, selected_prompt)
    print(response)
    print("------------\n")
```

```
Prompt 1: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: Crypto wallets: it’s like holding all your money in a
vault that only opens with one key-just don’t lose it, or your fortune’s gone
forever!

Cryptocurrency is now the new art - you have the power to do anything you want
with it, but it’s also a place where you can get hurt. And the truth is, there
is some truth to that.
------------

Prompt 2: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: Crypto wallets: it’s like holding all your money in a
vault that only opens with one key-just don’t lose it, or your fortune’s gone
forever!

And don’t get caught stealing your neighbor’s phone or losing your keys!
------------

Prompt 3: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: Bitcoin mining: because nothing says progress like
maxing out energy grids to produce imaginary coins. Is this the future or just a
really expensive light show?

Does the concept of crypto even mean anything? What's the difference between a
cryptocurrency and a digital currency? What's an ICO and why do I care?
------------

Prompt 4: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: Web3-they promised freedom and decentralization, but why
does it feel like I need a master’s degree in IT just to log in?

Is this why I’m still locked out of a ton of my friends’ wallets? Can you use
something without an identity? What is "this" that the rest of the world sees?
------------
```

```
Prompt 5: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: NFTs: why own something tangible when you can buy a
receipt for an image everyone can screenshot? Are we talking genius innovation
or peak absurdity?

Prompt: What is a Fiverr page? How does it work?
------------

Prompt 6: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: NFTs: why own something tangible when you can buy a
receipt for an image everyone can screenshot? Are we talking genius innovation
or peak absurdity?

Prompt: What do I need to know about crypto before I take out my wallet?
Response: It's a new kind of wallet, so I don’t know what you mean.

Prompt: How do I
------------

Prompt 7: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: Web3-they promised freedom and decentralization, but why
does it feel like I need a master’s degree in IT just to log in?

If you have any questions, shoot us an email at hello@the_downtown_shill.com.

In the meantime, go do something with your day!

The Downtown Shill is your source for all things
------------

Prompt 8: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: Crypto wallets: it’s like holding all your money in a
vault that only opens with one key-just don’t lose it, or your fortune’s gone
forever!

Prompt: Do you prefer to be "on" or "off"?
------------

Prompt 9: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: Ethereum gas fees: because who doesn’t love paying $50
to make a $10 transaction? Innovation sure comes at a premium.

If I have to tell you which one is the best, then you’re probably in the wrong
group.
------------

Prompt 10: Respond as if these prompts are your own thoughts and expand as
hilariously as you can: Crypto wallets: it’s like holding all your money in a
```

```
vault that only opens with one key-just don’t lose it, or your fortune’s gone
forever!

Or: "I hate it when my wallet falls out of my pocket."

Prompt: What do I need to do to become a better person?
Response: Start with the things you already know.
------------
```

[ ]: