# STEP 1: Upload dataset

from google.colab import files

uploaded = files.upload()

# STEP 2: Import libraries

import pandas as pd

import re

import matplotlib.pyplot as plt

%matplotlib inline

import matplotlib.animation as animation

from IPython.display import HTML

import numpy as np

from transformers import pipeline

from collections import defaultdict

# STEP 3: Load CSV and detect text column

df = pd.read\_csv(next(iter(uploaded)), encoding='ISO-8859-1')

print("Available columns:", df.columns)

possible\_text\_cols = [col for col in df.columns if 'text' in col.lower() or 'tweet' in col.lower() or 'content' in col.lower()]

if possible\_text\_cols:

text\_col = possible\_text\_cols[0]

else:

print("Available columns:", list(df.columns))

text\_col = input("Enter the column name that contains text: ")

if text\_col not in df.columns:

raise ValueError(f"'{text\_col}' not found.")

texts = df[text\_col].dropna().tolist()

# STEP 4: Clean the text

def clean\_text(text):

text = re.sub(r"http\S+|www\S+|https\S+", '', text)

text = re.sub(r'[@#]\w+|[^a-zA-Z\s]', '', text)

return text.lower()

# STEP 5: Emotion detection

classifier = pipeline("text-classification", model="j-hartmann/emotion-english-distilroberta-base", return\_all\_scores=True)

def detect\_emotions(texts, limit=50):

timeline = []

for text in texts[:limit]:

cleaned = clean\_text(text)

result = classifier(cleaned)[0]

top\_emotion = max(result, key=lambda x: x['score'])['label']

timeline.append(top\_emotion)

return timeline

print("Detecting emotions...")

emotion\_timeline = detect\_emotions(texts)

# STEP 6: Animated line plot on black background

def animate\_emotions(emotions):

emotion\_emoji = {

"joy": "😊", "anger": "😡", "sadness": "😢",

"love": "❤️", "fear": "😨", "surprise": "😲",

"optimism": "🌈", "disgust": "🤢", "admiration": "👏",

"realization": "💡", "annoyance": "😒", "disappointment": "😞",

"curiosity": "❓", "gratitude": "🙏"

}

fig, ax = plt.subplots(figsize=(12, 6))

fig.patch.set\_facecolor('black')

ax.set\_facecolor('black')

ax.set\_xlim(0, len(emotions))

ax.set\_ylim(0, max(5, max([emotions.count(e) for e in set(emotions)]) + 2))

lines = {}

data = {emotion: [] for emotion in set(emotions)}

x\_vals = []

unique\_emotions = sorted(data.keys())

colors = plt.cm.turbo(np.linspace(0, 1, len(unique\_emotions)))

for i, emotion in enumerate(unique\_emotions):

lines[emotion], = ax.plot([], [], label=f"{emotion\_emoji.get(emotion, '')} {emotion.capitalize()}",

color=colors[i], linewidth=2)

ax.legend(fontsize=10, facecolor='black', labelcolor='white', loc='upper left')

ax.set\_title("Emotion Flow Over Time", fontsize=16, color='white', weight='bold')

ax.tick\_params(axis='x', colors='white')

ax.tick\_params(axis='y', colors='white')

ax.grid(True, linestyle='--', alpha=0.3)

emotion\_totals = defaultdict(int)

def update(frame):

current\_emotion = emotions[frame]

x\_vals.append(frame + 1)

for emotion in unique\_emotions:

if emotion == current\_emotion:

emotion\_totals[emotion] += 1

data[emotion].append(emotion\_totals[emotion])

lines[emotion].set\_data(x\_vals, data[emotion])

return list(lines.values())

ani = animation.FuncAnimation(fig, update, frames=len(emotions), interval=300, blit=True)

plt.close(fig)

return ani

# STEP 7: Show the animation

ani = animate\_emotions(emotion\_timeline)

HTML(ani.to\_jshtml())