

#EMOTIONS DECODING

```
import torch

from transformers import AutoTokenizer, AutoModelForSequenceClassification

import torch.nn.functional as F

# Load pre-trained model and tokenizer for emotion analysis

model_name = "nateraw/bert-base-uncased-emotion"

tokenizer = AutoTokenizer.from_pretrained(model_name)

model = AutoModelForSequenceClassification.from_pretrained(model_name)

# List of emotions

emotions = ['admiration', 'amusement', 'anger', 'annoyance', 'approval',
            'caring', 'confusion', 'curiosity', 'desire', 'disappointment',
            'disapproval', 'disgust', 'embarrassment', 'excitement',
            'fear', 'gratitude', 'grief', 'joy', 'love', 'nervousness',
            'optimism', 'pride', 'realization', 'relief', 'remorse',
            'sadness', 'surprise', 'neutral']

def analyze_emotion(text):

    # Tokenize input text

    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)

    # Get model outputs

    with torch.no_grad():

        outputs = model(**inputs)

    # Convert logits to probabilities

    probs = F.softmax(outputs.logits, dim=1)
```

```
# Get top emotion
top_prob, top_class = torch.max(probs, dim=1)
emotion = emotions[top_class.item()]
```

```
return emotion, top_prob.item()
```

```
# Example usage
```

```
if __name__ == "__main__":
    print("Enter social media conversation (type 'exit' to quit):")
    while True:
        text = input(">> ")
        if text.lower() == "exit":
            break
        emotion, confidence = analyze_emotion(text)
        print(f"Detected Emotion: {emotion} (Confidence: {confidence:.2f})")
```

#MEAN EMOTIONS DEFECTION FROM MULTIPLE TEXTS

```
import torch

from transformers import AutoTokenizer, AutoModelForSequenceClassification

import torch.nn.functional as F

from collections import Counter


# Load model and tokenizer

model_name = "nateraw/bert-base-uncased-emotion"

tokenizer = AutoTokenizer.from_pretrained(model_name)

model = AutoModelForSequenceClassification.from_pretrained(model_name)


# Emotion labels

emotions = ['admiration', 'amusement', 'anger', 'annoyance', 'approval',
            'caring', 'confusion', 'curiosity', 'desire', 'disappointment',
            'disapproval', 'disgust', 'embarrassment', 'excitement',
            'fear', 'gratitude', 'grief', 'joy', 'love', 'nervousness',
            'optimism', 'pride', 'realization', 'relief', 'remorse',
            'sadness', 'surprise', 'neutral']


def analyze_emotion(text):

    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)

    with torch.no_grad():

        outputs = model(**inputs)

        probs = F.softmax(outputs.logits, dim=1)

        top_prob, top_class = torch.max(probs, dim=1)

        return emotions[top_class.item()]


def mean_emotion(text_list):
```

```
emotion_list = [analyze_emotion(text) for text in text_list]
emotion_count = Counter(emotion_list)
most_common = emotion_count.most_common(1)[0]
return most_common[0], emotion_count
```

Example usage

```
if __name__ == "__main__":
```

```
    texts = [
        "I love how this turned out!",
        "This makes me so angry!",
        "I'm feeling quite happy and grateful.",
        "Why did this happen to me?",
        "This is so annoying and unfair."
    ]
```

```
    mean_emotion_label, all_counts = mean_emotion(texts)
    print(f"Mean (most common) emotion: {mean_emotion_label}")
    print("All detected emotions:", dict(all_counts))
```

#BASIC SENTIMENT ANALYSIS USING TEXTLOB

```
from textblob import TextBlob
```

```
text = "I am really happy and excited about my future!"
```

```
blob = TextBlob(text)
```

```
sentiment = blob.sentiment
```

```
print("Text:", text)
```

```
print("Polarity:", sentiment.polarity)
```

```
print("Subjectivity:", sentiment.subjectivity)
```

#Emotion Detection Using Pretrained Transformer (Hugging Face)

```
import torch

from transformers import AutoTokenizer, AutoModelForSequenceClassification

import torch.nn.functional as F

tokenizer = AutoTokenizer.from_pretrained("nateraw/bert-base-uncased-emotion")

model = AutoModelForSequenceClassification.from_pretrained("nateraw/bert-base-uncased-emotion")

text = "I am feeling so grateful for your help."

inputs = tokenizer(text, return_tensors="pt")

outputs = model(**inputs)

probs = F.softmax(outputs.logits, dim=1)

labels = model.config.id2label

emotion = labels[probs.argmax().item()]

print("Text:", text)

print("Detected Emotion:", emotion)
```

#Mean Emotion from Multiple Sentences

```
from collections import Counter
```

```
texts = ["I'm so proud of you!", "This is annoying.", "I'm happy!", "Why am I so sad?", "Thank  
you!"]
```

```
emotion_list = []
```

```
for t in texts:
```

```
    inputs = tokenizer(t, return_tensors="pt")
```

```
    outputs = model(**inputs)
```

```
    probs = F.softmax(outputs.logits, dim=1)
```

```
    emotion = labels[probs.argmax().item()]
```

```
    emotion_list.append(emotion)
```

```
mean_emotion = Counter(emotion_list).most_common(1)[0][0]
```

```
print("Texts:", texts)
```

```
print("Detected Emotions:", emotion_list)
```

```
print("Most Common Emotion:", mean_emotion)
```

#Sentiment Classification Using Vader (Great for Social Media)

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()
text = "I can't believe how amazing this is!"

score = analyzer.polarity_scores(text)

print("Text:", text)
print("Scores:", score)
print("Sentiment:", "Positive" if score['compound'] > 0 else "Negative" if score['compound']
< 0 else "Neutral")
```


#Emotion Detection from a CSV of Comments

```
import pandas as pd

# Example comments
data = {'Comment': [
    "I am feeling sad.",
    "Wow, this made my day!",
    "You are so rude.",
    "Thanks a lot for helping me."
]}

df = pd.DataFrame(data)

# Detect emotion for each comment
df['Emotion'] = df['Comment'].apply(lambda x:
labels[torch.argmax(F.softmax(model(**tokenizer(x, return_tensors="pt"))).logits,
dim=1)).item()])

print(df)
```

#Real-Time Emotion Detection from User Input

```
while True:

    text = input("Enter your message (or 'exit'): ")

    if text.lower() == 'exit':

        break

    inputs = tokenizer(text, return_tensors="pt")

    outputs = model(**inputs)

    probs = F.softmax(outputs.logits, dim=1)

    emotion = labels[probs.argmax().item()]

    print(f"Detected Emotion: {emotion}")
```

#Detecting Top 3 Emotions with Probabilities

```
def top_emotions(text, top_k=3):  
    inputs = tokenizer(text, return_tensors="pt")  
    outputs = model(**inputs)  
    probs = F.softmax(outputs.logits, dim=1)[0]  
    top_probs, top_labels = torch.topk(probs, top_k)  
    for i in range(top_k):  
        print(f"{labels[top_labels[i].item()]}: {top_probs[i].item():.2f}")  
  
top_emotions("I'm feeling a mix of excitement, fear, and pride.")
```

Emotion Detection in YouTube Comments (Dummy Data)

```
comments = [  
    "This video is hilarious!",  
    "I didn't like the ending.",  
    "The song gave me chills.",  
    "I cried watching this."  
]  
  
for comment in comments:  
    inputs = tokenizer(comment, return_tensors="pt")  
    outputs = model(**inputs)  
    probs = F.softmax(outputs.logits, dim=1)  
    emotion = labels[probs.argmax().item()]  
    print(f"Comment: {comment} => Emotion: {emotion}")
```

#Emoji-Based Emotion Estimation

```
emoji_texts = [  
    "I love this! ❤️",  
    "I'm furious right now! ",  
    "Yay! ",  
    "Why me... ",  
]  
  
for text in emoji_texts:  
    inputs = tokenizer(text, return_tensors="pt")  
    outputs = model(**inputs)  
    emotion = labels[torch.argmax(F.softmax(outputs.logits, dim=1)).item()]  
    print(f"Text: {text} => Emotion: {emotion}")
```

Compare Emotions Between Two Comments

```
def compare_emotions(text1, text2):  
    def detect(text):  
        inputs = tokenizer(text, return_tensors="pt")  
        logits = model(**inputs).logits  
        return labels[torch.argmax(F.softmax(logits, dim=1)).item()]  
  
    e1 = detect(text1)  
    e2 = detect(text2)  
    print(f"Text 1: {text1} => {e1}")  
    print(f"Text 2: {text2} => {e2}")  
    print("Same emotion" if e1 == e2 else "Different emotions")  
  
compare_emotions("I'm scared about my exam", "I'm excited for the test!")
```