cleaned_analysis

September 18, 2025

[1]:
```
import os
import re
import nltk
import pandas as pd
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from collections import Counter
from transformers import pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
```

```
/Users/kanubalad/miniforge3/envs/migration/lib/python3.9/site-
packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

0.1 Packages and Libraries Installations

Install the following packages

- !pip install easyocr
- !sudo apt-get install ffmpeg
- !pip install openai-whisper (from openai but it needs the ffmeg too)

[2]:
```
nltk.download("stopwords")
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/kanubalad/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[3]:
```
def clean_text(text):
    text = text.lower()
    text = re.sub(r"[^a-z\s]", "", text)  # remove punctuation/numbers
    tokens = [w for w in text.split() if w not in stop_words]
```

```
    return " ".join(tokens)


def display_topics(model, feature_names, no_top_words):
    topics = {}
    for idx, topic in enumerate(model.components_):
        words = [feature_names[i] for i in topic.argsort()[:-no_top_words - 1:
 ↪-1]]
        topics[idx] = words
        print(f"\nTopic {idx+1}:")
        print(" | ".join(words))
    return topics


def get_top_ngrams(corpus, ngram_range=(2,2), n=20):
    vec = CountVectorizer(ngram_range=ngram_range, stop_words="english").
 ↪fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.
 ↪items()]
    words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
    return words_freq[:n]


def plot_ngrams(ngrams, title):
    phrases, freqs = zip(*ngrams)
    plt.figure(figsize=(10,5))
    plt.barh(phrases[::-1], freqs[::-1], color="skyblue")
    plt.title(title)
    plt.xlabel("Frequency")
    plt.ylabel("Phrases")
    plt.show()
```

[4]:
```
df = pd.read_csv("all_transcripts.csv")
df["cleaned"] = df["transcript"].astype(str).apply(clean_text)

all_words = " ".join(df["cleaned"]).split()
word_freq = Counter(all_words).most_common(10)
print("  Top 20 Most Frequent Words:")
for word, freq in word_freq:
    print(f"{word}: {freq}")
```

```
  Top 20 Most Frequent Words:
nurse: 41
ghana: 41
like: 40
```

```
im: 33
go: 33
come: 29
uk: 27
going: 26
get: 24
work: 24
```

0.1.1 Topic Modelling

[5]:
```
vectorizer = CountVectorizer(stop_words="english", max_features=5000)
X = vectorizer.fit_transform(df["cleaned"])
lda = LatentDirichletAllocation(n_components=5, random_state=42)
lda.fit(X)

topics = display_topics(lda, vectorizer.get_feature_names_out(), 10)
```

```
Topic 1:
nurse | uk | clinical | research | work | nursing | jobs | care | registered |
nmc

Topic 2:
nurses | ceo | company | visa | star | ghana | sponsor | ielts | like | watching

Topic 3:
like | ghana | come | going | im | got | nurse | want | work | need

Topic 4:
pain | okay | alright | im | ash | palm | dont | really | bang | oh

Topic 5:
tweed | oh | feel | speaks | language | wonderful | wala | illa | shes | care
```

0.2 Topic Modeling Interpretation (LDA Results)

This topic modeling was performed using Latent Dirichlet Allocation (LDA) with 5 topics, based on a corpus of cleaned text. Below is the interpretation of each topic based on the top 10 associated terms.

0.2.1 Topic 1: Nursing Careers & Healthcare Work (UK-focused)

Keywords: `nurse`, `uk`, `clinical`, `research`, `work`, `nursing`, `jobs`, `care`, `registered`, `nmc`

This topic centers around **professional nursing roles**, particularly within the **UK healthcare system**. The presence of terms like `nmc` (Nursing and Midwifery Council), `registered`, and `clinical` suggests discussions related to qualifications, job opportunities, and regulatory bodies in nursing.

0.2.2 Topic 2: Sponsorship, Migration & Career Aspirations

Keywords: `nurses`, `ceo`, `company`, `visa`, `star`, `ghana`, `sponsor`, `ielts`, `like`, `watching`

This topic likely reflects themes around **job sponsorship**, **immigration pathways**, and career aspirations, particularly for nurses from **Ghana** and similar contexts. Words like `visa`, `sponsor`, and `IELTS` point to the migration process, while `ceo` and `company` may relate to recruiters or organizations involved in this pipeline.

0.2.3 Topic 3: Personal Narratives & Migration Intent

Keywords: `like`, `ghana`, `come`, `going`, `im`, `got`, `nurse`, `want`, `work`, `need`

This is a **conversational topic** focused on **individual intent and aspiration**, especially among people in **Ghana** considering moving abroad to work, often in nursing. The language is informal, suggesting interviews, social media posts, or messaging data.

0.2.4 Topic 4: Expressions of Discomfort & Casual Dialogue

Keywords: `pain`, `okay`, `alright`, `im`, `ash`, `palm`, `dont`, `really`, `bang`, `oh`

This topic likely captures **emotional or physical discomfort** and casual, perhaps semi-scripted **dialogue or conversation snippets**. It may reflect responses from interviews, WhatsApp chats, or informal communication where people describe feelings or bodily experiences.

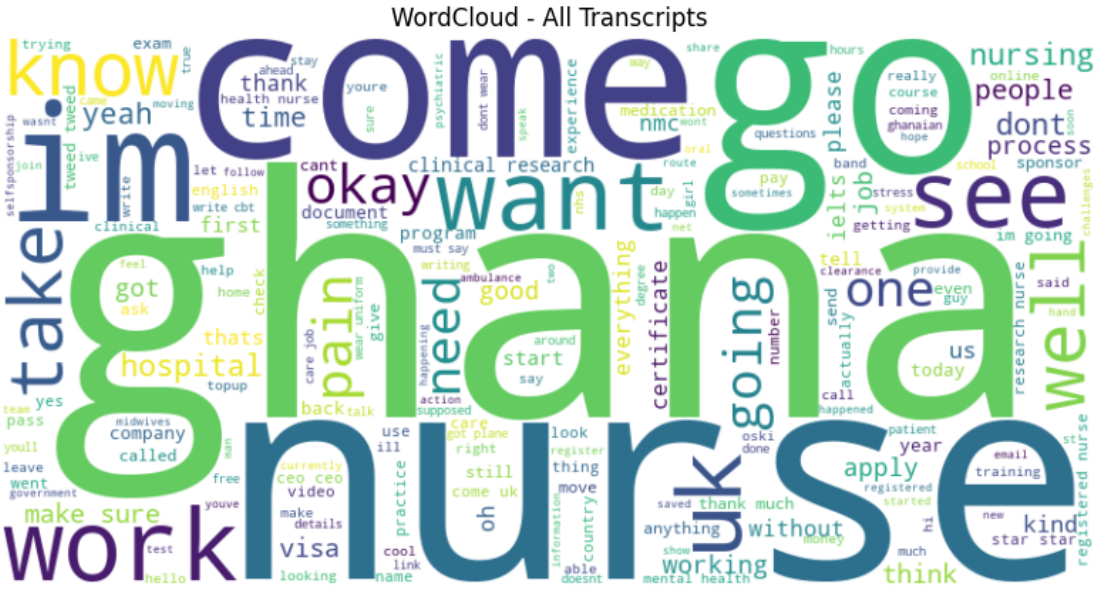
0.2.5 Topic 5: Language, Emotion & Identity Expression

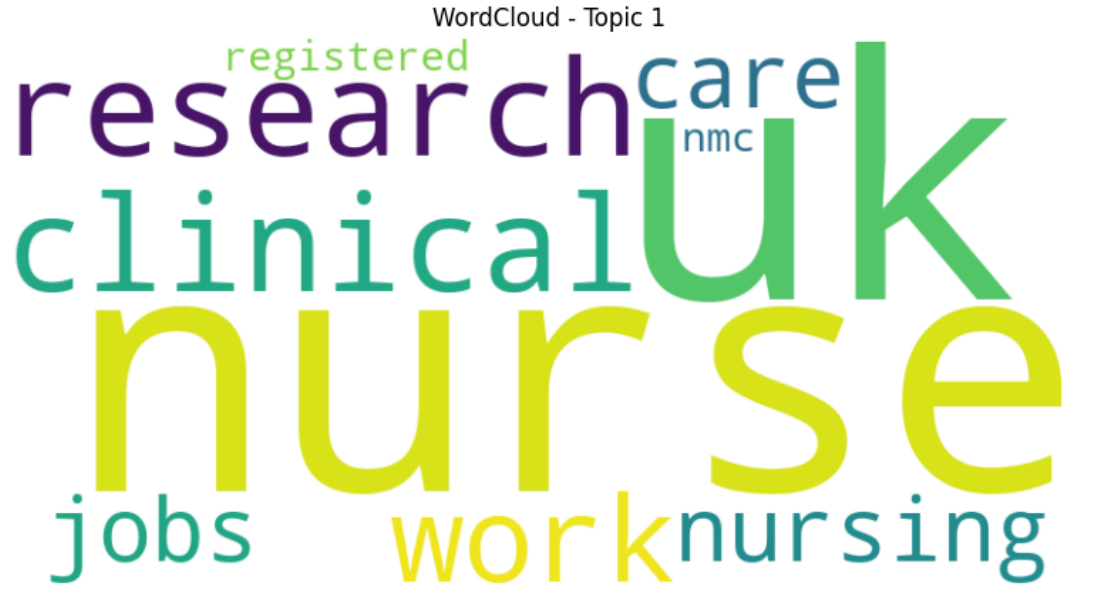
Keywords: `tweed`, `oh`, `feel`, `speaks`, `language`, `wonderful`, `wala`, `illa`, `shes`, `care`

This topic appears to reflect **language use, cultural identity, and emotional tone**. Words like `wala` and `illa` (possibly from regional dialects or expressions) suggest multilingual or localized speech patterns. The topic might relate to how participants express **care, emotion, or cultural connection**.

[6]:
```
wc_all = WordCloud(width=800, height=400, background_color="white").generate("␣
↪".join(df["cleaned"]))
plt.figure(figsize=(10, 5))
plt.imshow(wc_all, interpolation="bilinear")
plt.axis("off")
plt.title("WordCloud - All Transcripts")
plt.show()

# --- Step 6: Wordcloud per topic ---
for idx, words in topics.items():
```

```
    wc = WordCloud(width=800, height=400, background_color="white").generate("␣
↪".join(words))
    plt.figure(figsize=(10, 5))
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.title(f"WordCloud - Topic {idx+1}")
    plt.show()
```






WordCloud - Topic 2
sponsor
watching
star
visa
ceo
ghana
nurses
company
ielts


WordCloud - Topic 3
want
come
got
need
ghana
work
nurse
im
going

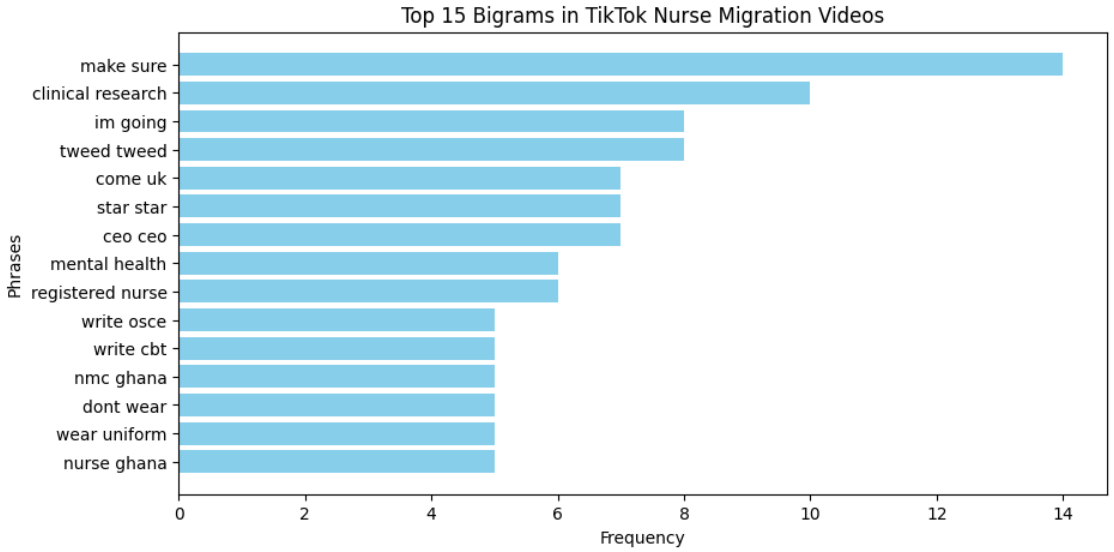
0.2.6 — Bigram & Trigram Analysis —

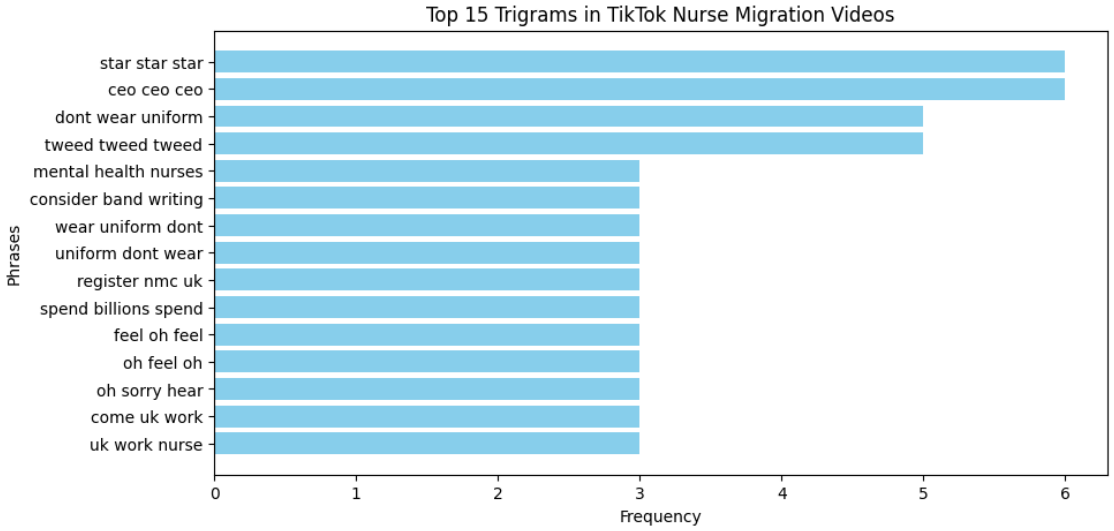
[7]:
```
bigrams = get_top_ngrams(df["cleaned"], ngram_range=(2,2), n=15)
trigrams = get_top_ngrams(df["cleaned"], ngram_range=(3,3), n=15)

print("\n Top Bigrams:", bigrams)
print("\n Top Trigrams:", trigrams)
```

```
# --- Plot results ---
plot_ngrams(bigrams, "Top 15 Bigrams in TikTok Nurse Migration Videos")
plot_ngrams(trigrams, "Top 15 Trigrams in TikTok Nurse Migration Videos")
```

Top Bigrams: [('make sure', np.int64(14)), ('clinical research', np.int64(10)), ('im going', np.int64(8)), ('tweed tweed', np.int64(8)), ('come uk', np.int64(7)), ('star star', np.int64(7)), ('ceo ceo', np.int64(7)), ('mental health', np.int64(6)), ('registered nurse', np.int64(6)), ('write osce', np.int64(5)), ('write cbt', np.int64(5)), ('nmc ghana', np.int64(5)), ('dont wear', np.int64(5)), ('wear uniform', np.int64(5)), ('nurse ghana', np.int64(5))]

Top Trigrams: [('star star star', np.int64(6)), ('ceo ceo ceo', np.int64(6)), ('dont wear uniform', np.int64(5)), ('tweed tweed tweed', np.int64(5)), ('mental health nurses', np.int64(3)), ('consider band writing', np.int64(3)), ('wear uniform dont', np.int64(3)), ('uniform dont wear', np.int64(3)), ('register nmc uk', np.int64(3)), ('spend billions spend', np.int64(3)), ('feel oh feel', np.int64(3)), ('oh feel oh', np.int64(3)), ('oh sorry hear', np.int64(3)), ('come uk work', np.int64(3)), ('uk work nurse', np.int64(3))]





0.3 Sentiment Analysis

[8]:
```
sentiment_analyzer = pipeline("sentiment-analysis")
df["sentiment"] = df["transcript"].astype(str).apply(lambda x:␣
 ↪sentiment_analyzer(x[:512])[0]["label"])
emotion_analyzer = pipeline("text-classification",
                            model="j-hartmann/
 ↪emotion-english-distilroberta-base",
                            return_all_scores=False)
df["emotion"] = df["transcript"].astype(str).apply(lambda x:␣
 ↪emotion_analyzer(x[:512])[0]["label"])
df.to_csv("transcripts_with_sentiment_emotion.csv", index=False)
```

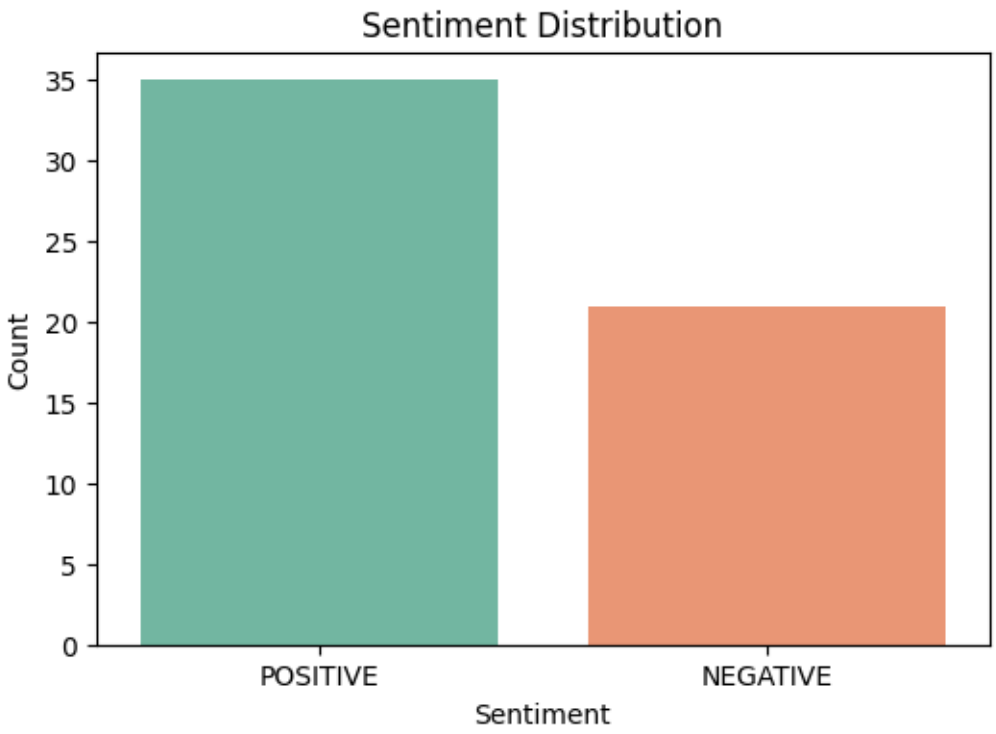
```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-
finetuned-sst-2-english and revision 714eb0f
(https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-
sst-2-english).
Using a pipeline without specifying a model name and revision in production is
not recommended.
Device set to use mps:0
Device set to use mps:0
/Users/kanubalad/miniforge3/envs/migration/lib/python3.9/site-
packages/transformers/pipelines/text_classification.py:111: UserWarning:
`return_all_scores` is now deprecated,  if want a similar functionality use
`top_k=None` instead of `return_all_scores=True` or `top_k=1` instead of
`return_all_scores=False`.
  warnings.warn(
```

0.3.1 Analysis and Plots

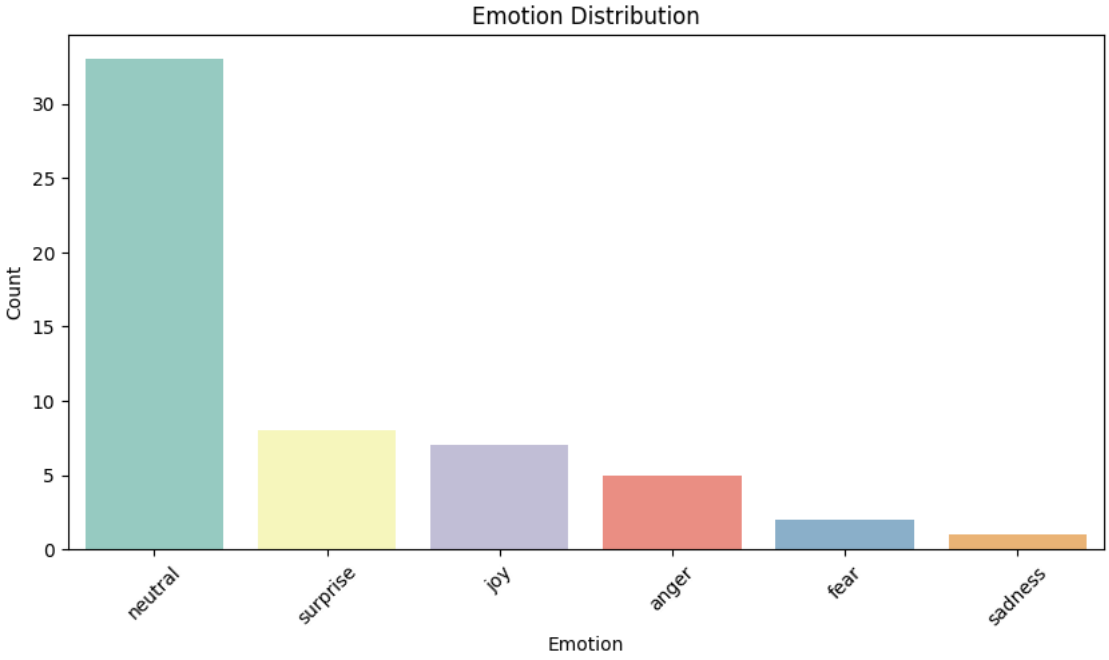
[9]:
```
df = pd.read_csv("transcripts_with_sentiment_emotion.csv")
```

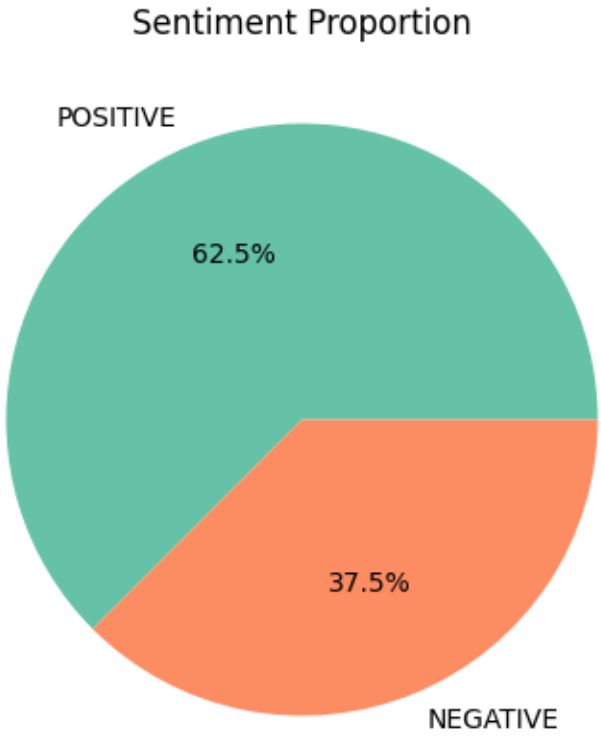
[10]:
```
# --- Sentiment distribution ---
plt.figure(figsize=(6,4))
sns.countplot(x="sentiment", data=df, palette="Set2")
plt.title("Sentiment Distribution")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.show()

# --- Emotion distribution ---
plt.figure(figsize=(10,5))
sns.countplot(x="emotion", data=df, order=df["emotion"].value_counts().index,␣
 ↪palette="Set3")
plt.title("Emotion Distribution")
plt.xlabel("Emotion")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()

# --- Proportions (optional pie chart) ---
sentiment_counts = df["sentiment"].value_counts(normalize=True)
emotion_counts = df["emotion"].value_counts(normalize=True)

plt.figure(figsize=(5,5))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct="%1.1f%%",␣
 ↪colors=sns.color_palette("Set2"))
plt.title("Sentiment Proportion")
plt.show()

plt.figure(figsize=(6,6))
plt.pie(emotion_counts, labels=emotion_counts.index, autopct="%1.1f%%",␣
 ↪colors=sns.color_palette("Set3"))
plt.title("Emotion Proportion")
plt.show()
```

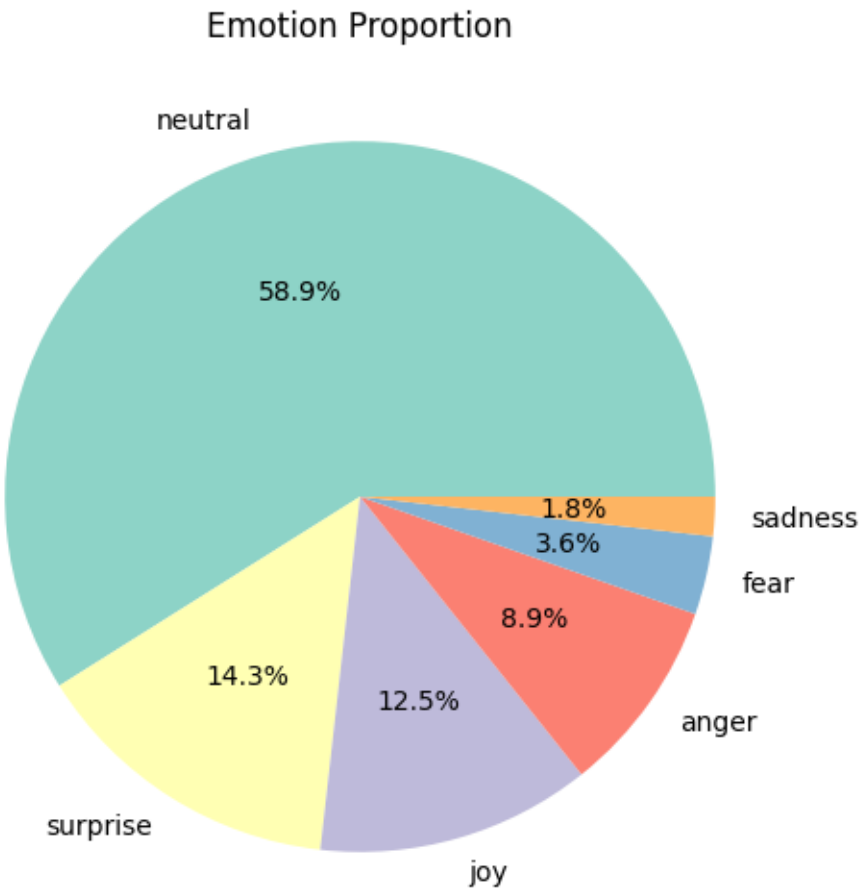
```
/var/folders/cf/xgy1fywj3q1bylrhvlp36psc0000gn/T/ipykernel_59582/2522955112.py:3
: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x="sentiment", data=df, palette="Set2")
```




```
/var/folders/cf/xgy1fywj3q1bylrhvlp36psc0000gn/T/ipykernel_59582/2522955112.py:1
1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x="emotion", data=df, order=df["emotion"].value_counts().index,
palette="Set3")
```


Emotion Distribution
Count
30
25
20
15
10
5
0
neutral
surprise
joy
anger
fear
sadness
Emotion


Sentiment Proportion
POSITIVE
62.5%
37.5%
NEGATIVE


Emotion Proportion
neutral
58.9%
sadness
1.8%
fear
3.6%
anger
8.9%
joy
12.5%
surprise
14.3%