**ChatGPT Solution No.1:**

Hello ChatGPT I am a beginner digital humanities student and I want you to help me with this problem so I have a folder in my directory named data where in data I have a folder of articles now what I want to do is to filter the articles using a for loop in python that were published in January 2024 and october 2023. So the articles filename start with this format year\_month\_date\_a unique code for the article I know the filenaming will help with the filtering can you help me do this

ChatGPT said:

Absolutely, I can help you with that! Since the article filenames follow the pattern:

css

Copy

Edit

year\_month\_day\_unique-code.txt

We can use this structure to filter filenames that start with 2024\_01\_ (for January 2024) or 2023\_10\_ (for October 2023).

Here's a simple Python script that does this using a for loop:

✅ Python Script

python

Copy

Edit

import os

# Path to your articles folder

articles\_path = "data/articles"

# Target prefixes for filtering

target\_prefixes = ["2024\_01\_", "2023\_10\_"]

# Loop through the files in the folder

for filename in os.listdir(articles\_path):

# Check if the filename starts with one of the target prefixes

if any(filename.startswith(prefix) for prefix in target\_prefixes):

print(filename)

🔍 How it works:

os.listdir() gets all filenames in the articles folder.

The any() function checks if the filename starts with either of the target prefixes.

Matching filenames are printed (you can modify this to do other operations too, like opening or analyzing them).

If you want to store these filenames in a list instead of printing them, just do:

python

Copy

Edit

filtered\_files = []

for filename in os.listdir(articles\_path):

if any(filename.startswith(prefix) for prefix in target\_prefixes):

filtered\_files.append(filename)

print(filtered\_files)

Let me know if your files have extensions like .txt or if you want to do something more with the filtered articles (like open and read them).

You said:

okay so we are creating the empty list named filtered\_files to store our filtered articles published in jan 2024 and oct 2023

ChatGPT said:

Exactly! The empty list filtered\_files will hold all filenames that match your criteria (published in January 2024 or October 2023). As the loop goes through each file in the folder, it checks if the filename starts with either "2024\_01\_" or "2023\_10\_". If it does, that filename gets added to filtered\_files.

After the loop finishes, filtered\_files will contain all your filtered article filenames.

If you want, I can help you write the full code with comments or add the next step—like opening and reading those files. Would you like to do that?

**ChatGPT Solution No.2:**

So now we filtered the articles published in Jan 2024 and October 2023 the next thing I want to do is to split the title and body of each article because I think title is not useful for the sentiment analysis also at the end of sentiment analysis I am creating a csv file where title will be one of column so I want the title for that as well can you help me with this

ChatGPT said:

Got it! You want to:

Filter the articles by filename (which we did).

For each filtered article, separate title and body.

Use body for sentiment analysis.

Keep title to save later in a CSV file.

Here’s a step-by-step approach:

Assumptions:

Each article file contains a title and body inside the text file.

The title and body are separated by a line break (for example, the first line is the title, and the rest is the body).

Files are text files (.txt).

Code snippet to do this:

python

Copy

Edit

import os

import csv

articles\_path = "data/articles"

target\_prefixes = ["2024\_01\_", "2023\_10\_"]

filtered\_files = []

# Prepare a list to hold data for CSV: each item will be a dict with 'title' and 'body'

articles\_data = []

for filename in os.listdir(articles\_path):

if any(filename.startswith(prefix) for prefix in target\_prefixes):

filtered\_files.append(filename)

# Open the article file and read contents

filepath = os.path.join(articles\_path, filename)

with open(filepath, 'r', encoding='utf-8') as file:

lines = file.readlines()

# Assume first line is title, rest is body

title = lines[0].strip()

body = "".join(lines[1:]).strip()

# Save title and body for later sentiment analysis and CSV creation

articles\_data.append({

"title": title,

"body": body

})

# Example: Print titles and first 100 chars of body to verify

for article in articles\_data:

print("Title:", article["title"])

print("Body snippet:", article["body"][:100])

print("---")

# At this point you can run sentiment analysis on article["body"]

# and keep article["title"] for your CSV file later

How to save the results into a CSV with title and sentiment?

Here’s a quick example to write title and a dummy sentiment score:

python

Copy

Edit

# Let's assume you have sentiment results stored in articles\_data with a new key 'sentiment'

# For example, dummy sentiment:

for article in articles\_data:

article["sentiment"] = "positive" # Replace this with your real sentiment result

# Write to CSV

csv\_filename = "sentiment\_results.csv"

with open(csv\_filename, 'w', newline='', encoding='utf-8') as csvfile:

fieldnames = ["title", "sentiment"]

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

writer.writeheader()

for article in articles\_data:

writer.writerow({"title": article["title"], "sentiment": article["sentiment"]})

print(f"CSV saved as {csv\_filename}")

If you want, I can help you integrate sentiment analysis or give tips on libraries you can use for that. Would you like that?

You said:

okay so I forgot to tell you something that in our articles body and title is separated by this "-----" dashes

ChatGPT said:

Thanks for letting me know! Since the title and body are separated by "-----" dashes inside the article file, we can split the file content on that delimiter instead of assuming line breaks.

Here’s how you can update the reading part to split using "-----":

python

Copy

Edit

import os

articles\_path = "data/articles"

target\_prefixes = ["2024\_01\_", "2023\_10\_"]

filtered\_files = []

articles\_data = []

for filename in os.listdir(articles\_path):

if any(filename.startswith(prefix) for prefix in target\_prefixes):

filtered\_files.append(filename)

filepath = os.path.join(articles\_path, filename)

with open(filepath, 'r', encoding='utf-8') as file:

content = file.read()

# Split by '-----' to separate title and body

parts = content.split('-----')

if len(parts) >= 2:

title = parts[0].strip()

body = parts[1].strip()

else:

# If no '-----' found, treat whole content as body and empty title

title = ""

body = content.strip()

articles\_data.append({

"title": title,

"body": body

})

# Verify

for article in articles\_data:

print("Title:", article["title"])

print("Body snippet:", article["body"][:100])

print("---")

What changed?

Instead of reading lines, we read the whole content at once with file.read().

Then we split on the string "-----".

The first part is the title, the second part is the body.

If the delimiter is missing, it treats the entire content as the body and leaves the title empty (just in case).

Would you like me to help with the next step, like integrating sentiment analysis or saving to CSV?

You said:

can we simply loop through the filtered articles and then read them and split the body and title so for the moment help me do that

ChatGPT said:

Absolutely! Here's a simple, clear code snippet that loops through the filtered article filenames, reads each file, and splits the title and body by "-----" for you:

python

Copy

Edit

import os

articles\_path = "data/articles"

target\_prefixes = ["2024\_01\_", "2023\_10\_"]

filtered\_files = []

for filename in os.listdir(articles\_path):

if any(filename.startswith(prefix) for prefix in target\_prefixes):

filtered\_files.append(filename)

# Now loop through filtered articles, read and split title and body

for filename in filtered\_files:

filepath = os.path.join(articles\_path, filename)

with open(filepath, 'r', encoding='utf-8') as file:

content = file.read()

parts = content.split('-----')

if len(parts) >= 2:

title = parts[0].strip()

body = parts[1].strip()

else:

title = ""

body = content.strip()

print(f"Filename: {filename}")

print(f"Title: {title}")

print(f"Body (first 100 chars): {body[:100]}")

print('---')

This code:

Filters your files by prefix first,

Then reads each filtered article,

Splits the content into title and body by the "-----" delimiter,

Prints out the filename, title, and a snippet of the body.

You can build on this later for sentiment analysis or saving data. Want me to help with that next?

You said:

if len(parts) >= 2:

I think we donot need this len function as well the else statement and also the printing at the end

ChatGPT said:

Got it! If you’re confident every article has the "-----" separator, you can simplify by removing the len(parts) check and the else block. Also, no printing at the end if you don’t need it.

Here’s the cleaner version:

python

Copy

Edit

import os

articles\_path = "data/articles"

target\_prefixes = ["2024\_01\_", "2023\_10\_"]

filtered\_files = []

for filename in os.listdir(articles\_path):

if any(filename.startswith(prefix) for prefix in target\_prefixes):

filtered\_files.append(filename)

for filename in filtered\_files:

filepath = os.path.join(articles\_path, filename)

with open(filepath, 'r', encoding='utf-8') as file:

content = file.read()

title, body = content.split('-----', 1)

title = title.strip()

body = body.strip()

This will:

Assume every file contains exactly one "-----" separator,

Split into title and body accordingly,

Strip any extra whitespace.

If you want, I can help you store these in a list or dictionary for further use.

You said:

, 1)

now why do we need this one

ChatGPT said:

Great question! The 1 in this line:

python

Copy

Edit

title, body = content.split('-----', 1)

is the maxsplit parameter in Python’s str.split() method. Here’s why it matters:

content.split('-----', 1) tells Python:

Split the string at the first occurrence of '-----' only.

Without the 1, if your article body contains '-----' somewhere else (maybe as part of the text), the split would break the text into more than two parts, causing an error when you try to unpack into exactly two variables (title and body).

Using maxsplit=1 ensures:

You get exactly two parts:

Everything before the first '-----' → title

Everything after the first '-----' → body (including any further '-----' inside the body)

Quick example:

python

Copy

Edit

text = "My Title ----- This is the body ----- with extra dashes"

print(text.split('-----'))

# Output: ['My Title ', ' This is the body ', ' with extra dashes']

print(text.split('-----', 1))

# Output: ['My Title ', ' This is the body ----- with extra dashes']

Summary:

The 1 prevents splitting at all dashes, only the first one, which keeps your title/body separation safe and predictable. Without it, unpacking could fail or produce wrong results.

Let me know if you want me to explain this with any other examples!

You said:

okay understood

ChatGPT said:

Awesome! Glad that made sense. If you want, we can move on to the next step—like doing the sentiment analysis on the article bodies or saving your data to CSV. Just let me know!

**ChatGPT Solution No. 3:**

Okay so now we have splited the articles into body and titlle next I want to do sentiment analysis on the body I know first I have to give the body to the nlp processor then to create list to collect sentiment scores for each sentence 0 as negative 1 as neutral and 2 as positive and then to loop through all sentences in the article body to analyze sentiment individually and store those scores in the sentiments list I have created can you help me doing these tasks

ChatGPT said:

Yes! You're thinking about sentiment analysis in exactly the right sequence. Since you're working in Python and want to analyze each sentence in the article body, we’ll use a pre-trained NLP model for sentiment analysis. The easiest way is using Hugging Face's transformers library with a model like cardiffnlp/twitter-roberta-base-sentiment, which outputs 0 = negative, 1 = neutral, 2 = positive — just as you want.

✅ Step-by-step plan for you:

1. Install necessary libraries (if not already installed)

If you're on Jupyter or Colab, you can run:

bash

Copy

Edit

pip install transformers

pip install nltk

2. Import everything you need

python

Copy

Edit

from transformers import AutoTokenizer, AutoModelForSequenceClassification

from transformers import pipeline

import nltk

nltk.download('punkt') # for sentence tokenization

from nltk.tokenize import sent\_tokenize

3. Load the sentiment analysis model

python

Copy

Edit

model\_name = "cardiffnlp/twitter-roberta-base-sentiment"

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForSequenceClassification.from\_pretrained(model\_name)

nlp = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer)

✅ Now apply this to your article bodies

Here’s how to tokenize sentences, analyze each one, and store scores:

python

Copy

Edit

for filename in filtered\_files:

filepath = os.path.join(articles\_path, filename)

with open(filepath, 'r', encoding='utf-8') as file:

content = file.read()

title, body = content.split('-----', 1)

title = title.strip()

body = body.strip()

# Split the body into sentences

sentences = sent\_tokenize(body)

# Initialize list to hold sentiment scores (0, 1, or 2)

sentiment\_scores = []

# Analyze each sentence

for sentence in sentences:

result = nlp(sentence)[0] # Get result like {'label': 'LABEL\_2', 'score': 0.95}

label = result['label']

# Convert label to score

if label == 'LABEL\_0':

sentiment\_scores.append(0) # Negative

elif label == 'LABEL\_1':

sentiment\_scores.append(1) # Neutral

elif label == 'LABEL\_2':

sentiment\_scores.append(2) # Positive

print(f"Title: {title}")

print(f"Sentiment scores: {sentiment\_scores}")

print('---')

📌 Summary:

You read and split each article.

Broke the body into sentences.

Analyzed sentiment per sentence.

Stored sentiment results as a list of numbers (0, 1, 2).

Would you like to:

Calculate overall sentiment per article (e.g. majority)?

Save title + sentiment summary to CSV?

Let me know!

You said:

I am doing it on stanza and already have done the rest of the tasks I want you to help me what I have asked

ChatGPT said:

Perfect! Since you're using Stanza (and you've already set up the NLP pipeline and completed earlier steps), let’s go directly to what you asked:

✅ You want to:

Send the article body to Stanza’s NLP processor.

Tokenize it into sentences.

Create a list to store sentiment scores for each sentence.

Loop through each sentence, analyze its sentiment, and store the scores as 0 (neg), 1 (neutral), 2 (positive).

✅ Here's the code that does exactly that with Stanza:

python

Copy

Edit

import stanza

# Make sure you have the English model downloaded

# stanza.download('en') # Run this only once

# Initialize the NLP pipeline

nlp = stanza.Pipeline(lang='en', processors='tokenize,sentiment')

# Loop through filtered articles

for filename in filtered\_files:

filepath = os.path.join(articles\_path, filename)

with open(filepath, 'r', encoding='utf-8') as file:

content = file.read()

title, body = content.split('-----', 1)

title = title.strip()

body = body.strip()

# Process the article body with Stanza NLP

doc = nlp(body)

# Create a list to store sentiment scores

sentiment\_scores = []

# Loop through each sentence and extract sentiment

for sentence in doc.sentences:

sentiment = sentence.sentiment # 0 = negative, 1 = neutral, 2 = positive

sentiment\_scores.append(sentiment)

# (Optional) Store results in a dictionary or list if needed

print(f"Title: {title}")

print(f"Sentiment Scores: {sentiment\_scores}")

print('---')

📘 Quick Notes:

stanza.Pipeline(lang='en', processors='tokenize,sentiment') is all you need.

sentence.sentiment gives you:

0 → Negative

1 → Neutral

2 → Positive

No need to tokenize manually—Stanza does it automatically.

Would you like to compute an overall sentiment per article next (e.g. by average or majority vote)? Or move on to saving the results in a CSV?

**ChatGPT Solution No. 4:**

okay so I filtered the articles splited the title and body and then did the sentiment analysis for each sentence and then calculate the average sentiment and rounded off the average sentiment upto 2 decimal places so I am done with I had to do but I am remaining with creating a list named results where I will place my end results like filename, year\_month, title and then avg\_sentiment also after that I want to create a csv file named avg\_sentiment\_results.csv where I want to have these four columns filename, year\_month, title and then avg\_sentiment with data

ChatGPT said:

Excellent progress! You're almost done 🎉

Now let's wrap everything up step by step:

✅ 1. Create a results list with this structure:

Each result will be a dictionary with:

"filename" → the article file name

"year\_month" → from the file name (e.g. "2024\_01")

"title" → extracted from article

"avg\_sentiment" → rounded to 2 decimal places

✅ 2. Save the results to a CSV file named avg\_sentiment\_results.csv

✅ Final Code:

python

Copy

Edit

import os

import csv

import stanza

# Make sure English models are downloaded

# stanza.download('en')

# Initialize NLP

nlp = stanza.Pipeline(lang='en', processors='tokenize,sentiment')

# File path setup

articles\_path = "data/articles"

target\_prefixes = ["2024\_01\_", "2023\_10\_"]

filtered\_files = []

# Filter files

for filename in os.listdir(articles\_path):

if any(filename.startswith(prefix) for prefix in target\_prefixes):

filtered\_files.append(filename)

# Initialize final results list

results = []

# Process each filtered article

for filename in filtered\_files:

filepath = os.path.join(articles\_path, filename)

with open(filepath, 'r', encoding='utf-8') as file:

content = file.read()

title, body = content.split('-----', 1)

title = title.strip()

body = body.strip()

# NLP processing

doc = nlp(body)

# Sentiment scores per sentence

sentiment\_scores = [sentence.sentiment for sentence in doc.sentences]

# Average sentiment

if sentiment\_scores:

avg\_sentiment = round(sum(sentiment\_scores) / len(sentiment\_scores), 2)

else:

avg\_sentiment = None # or 1.0 (neutral) if you prefer

# Extract year\_month from filename

year\_month = "\_".join(filename.split("\_")[:2])

# Append to results

results.append({

"filename": filename,

"year\_month": year\_month,

"title": title,

"avg\_sentiment": avg\_sentiment

})

# Write results to CSV

csv\_file = "avg\_sentiment\_results.csv"

with open(csv\_file, mode='w', encoding='utf-8', newline='') as f:

writer = csv.DictWriter(f, fieldnames=["filename", "year\_month", "title", "avg\_sentiment"])

writer.writeheader()

writer.writerows(results)

print("CSV file created successfully: avg\_sentiment\_results.csv")

✅ Output:

You will get a CSV file named avg\_sentiment\_results.csv with columns:

filename year\_month title avg\_sentiment

2024\_01\_05\_xyz.txt 2024\_01 Some Article Title 1.33

Let me know if you want to also add sentiment category (like “Positive”, “Neutral”, “Negative”) based on average!

You said:

thank you so much for helping out me with creating my own data frame

ChatGPT said:

You're very welcome! 😊

You did an amazing job walking through every step—from filtering files to performing sentiment analysis and finally organizing your results into a DataFrame/CSV. That's a full mini project in digital humanities and NLP! 🚀

If you ever want to:

Visualize your sentiment trends,

Add more metadata (e.g. article length),

Compare sentiments across months,

Or train your own model on this data…

I’m here to help. Happy coding and analyzing! 📊🧠✨

**ChatGPT Solution No. 5:**

I'm getting a NameError for filename in my loop when I'm trying to extract year\_month from the article filename. The filenames look like this: 2023-10-01-1234.txt. How can I resolve this issue and also give me response in a way that I should understand

ChatGPT said:

Absolutely — let’s walk through it clearly and simply.

🛑 Problem:

You're getting this error:

pgsql

Copy

Edit

NameError: name 'filename' is not defined

This means:

👉 You’re trying to use the variable filename, but Python doesn’t know what it is — because it hasn’t been defined in that part of your code.

🧠 Why is this happening?

Usually, this happens when you write something like this:

python

Copy

Edit

for filename in filtered\_files:

# do some stuff

# Here you're outside the loop

year\_month = filename[:7] # ❌ This will cause a NameError

In the line above, filename only exists inside the for loop. So when you try to use it after the loop, it doesn’t exist anymore.

✅ How to fix it?

You should move your year\_month = ... line inside the loop, like this:

python

Copy

Edit

for filename in filtered\_files:

filepath = os.path.join(articles\_path, filename)

with open(filepath, 'r', encoding='utf-8') as file:

content = file.read()

title, body = content.split('-----', 1)

title = title.strip()

body = body.strip()

doc = nlp(body)

sentiment\_scores = [sentence.sentiment for sentence in doc.sentences]

avg\_sentiment = round(sum(sentiment\_scores) / len(sentiment\_scores), 2) if sentiment\_scores else None

# ✅ Clean and safe way to get year and month from filename

year\_month = filename[:7] # gets '2023-10' from '2023-10-01-1234.txt'

# ✅ Add everything to results

results.append([filename, year\_month, title, avg\_sentiment])

💡 Why filename[:7] works?

Let’s break it down with this example filename:

2023-10-01-1234.txt

python

Copy

Edit

filename[:7] → '2023-10'

It just picks the first 7 characters — which gives you the year and month you need. It’s clean, short, and doesn't rely on splitting strings.

✅ In summary:

Fix the NameError by making sure filename is used inside the for loop.

Use filename[:7] to cleanly extract year-month.

Append your result inside the loop so that all values are in scope.

Let me know if you want to turn this into a reusable function or notebook cell template!

**ChatGPT Solution No. 6:**

Hello ChatGPT so I have csv file named avg\_sentiment\_results.csv which contains four columns named filename, year\_month, title, avg\_sentiment. I have loaded it as panda dataframe on the python and now I want to create a bar chart out of it but what I want to do first is to group together articles that avg-sentiment from 0-0.5 as negative, 0.5-1.5 as negative and 1.5-2 as positive for the bar chart I want the sentiments as positive, negative, and neutral on x axis and count on y axis can you help me how can I do this guide me as a tutor

ChatGPT said:

Absolutely, I’ll guide you step-by-step like a tutor. Let’s assume you already loaded your CSV file into a pandas DataFrame. We'll go through the process in parts:

📘 Step 1: Import Required Libraries

First, make sure you’ve imported the necessary libraries:

python

Copy

Edit

import pandas as pd

import matplotlib.pyplot as plt

📗 Step 2: Load the CSV

Since you’ve already loaded it, it might look like this:

python

Copy

Edit

df = pd.read\_csv('avg\_sentiment\_results.csv')

You can preview the DataFrame to make sure it loaded correctly:

python

Copy

Edit

print(df.head())

📙 Step 3: Categorize Sentiments

Let’s create a new column called sentiment\_label based on avg\_sentiment. According to your rules:

0 to 0.5 → Negative

0.5 to 1.5 → Neutral

1.5 to 2 → Positive

Here’s how you can do it:

python

Copy

Edit

def label\_sentiment(score):

if 0 <= score < 0.5:

return 'Negative'

elif 0.5 <= score < 1.5:

return 'Neutral'

elif 1.5 <= score <= 2:

return 'Positive'

else:

return 'Unknown' # just in case any value is out of range

df['sentiment\_label'] = df['avg\_sentiment'].apply(label\_sentiment)

🧠 This function uses apply() to classify each score.

📕 Step 4: Count Articles in Each Category

Now count how many articles fall into each sentiment label:

python

Copy

Edit

sentiment\_counts = df['sentiment\_label'].value\_counts()

print(sentiment\_counts)

You’ll get a Series like:

mathematica

Copy

Edit

Neutral 100

Positive 50

Negative 40

To sort them in a desired order for plotting:

python

Copy

Edit

sentiment\_counts = sentiment\_counts.reindex(['Positive', 'Neutral', 'Negative'])

📒 Step 5: Plot the Bar Chart

Now let's plot the bar chart:

python

Copy

Edit

plt.figure(figsize=(8, 5))

sentiment\_counts.plot(kind='bar', color=['green', 'orange', 'red'])

plt.title('Sentiment Distribution of Articles')

plt.xlabel('Sentiment')

plt.ylabel('Number of Articles')

plt.xticks(rotation=0)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

✅ Summary of What You Did

Loaded your CSV data.

Defined sentiment categories based on numerical score.

Counted articles in each sentiment category.

Plotted a clean bar chart.

Let me know if you’d like to:

Save this chart as an image.

Show percentages instead of counts.

Customize colors and labels further.

Want to try each step in your Python environment and let me know where you are?

You said:

I want to create the bar chart using plotly express not matplotlib.pyplot

ChatGPT said:

Great choice! Plotly Express makes beautiful and interactive charts. I’ll guide you step by step again, now using Plotly Express instead of matplotlib.pyplot.

✅ Step-by-Step Guide Using Plotly Express

📘 Step 1: Import Required Libraries

python

Copy

Edit

import pandas as pd

import plotly.express as px

📗 Step 2: Load Your CSV File

(Since you’ve already done this, this is just for reference)

python

Copy

Edit

df = pd.read\_csv('avg\_sentiment\_results.csv')

📙 Step 3: Categorize Sentiments

We’ll define a function and apply it to a new column:

python

Copy

Edit

def label\_sentiment(score):

if 0 <= score < 0.5:

return 'Negative'

elif 0.5 <= score < 1.5:

return 'Neutral'

elif 1.5 <= score <= 2:

return 'Positive'

else:

return 'Unknown' # fallback for out-of-range values

df['sentiment\_label'] = df['avg\_sentiment'].apply(label\_sentiment)

📕 Step 4: Count Articles Per Sentiment

Plotly Express can group and count for you, but it’s usually clearer to prepare a separate summary:

python

Copy

Edit

sentiment\_counts = df['sentiment\_label'].value\_counts().reindex(['Positive', 'Neutral', 'Negative']).reset\_index()

sentiment\_counts.columns = ['Sentiment', 'Count']

Now sentiment\_counts looks like:

Sentiment Count

Positive 50

Neutral 100

Negative 40

📒 Step 5: Create the Bar Chart Using Plotly Express

python

Copy

Edit

fig = px.bar(

sentiment\_counts,

x='Sentiment',

y='Count',

color='Sentiment',

color\_discrete\_map={

'Positive': 'green',

'Neutral': 'orange',

'Negative': 'red'

},

title='Sentiment Distribution of Articles',

labels={'Sentiment': 'Sentiment Category', 'Count': 'Number of Articles'},

text='Count'

)

fig.update\_traces(textposition='outside')

fig.update\_layout(

xaxis\_title='Sentiment',

yaxis\_title='Number of Articles',

uniformtext\_minsize=8,

uniformtext\_mode='hide'

)

fig.show()

💡 What’s Great About Plotly?

Hover tooltips

Easy export to HTML or PNG

Colors and layout are highly customizable

Would you like to:

Add year/month filters or interactivity?

Display percentages instead of raw counts?

Let me know how you'd like to extend it!

You said:

also I forgot to add one thing more I also want separate bar charts for two months because the csv file contains data related to two different months October 2023 and January 2025 the column named year\_month has mentions the month and year it were published I want a separate bar chart for Jan 2024 and October 2023 also I want a simple map donot give me codes to make it fancier

ChatGPT said:

Perfect! Let’s keep it simple and focus on separate bar charts for each month using Plotly Express.

✅ Step-by-Step with Two Separate Bar Charts

Assuming your column year\_month contains values like "2023-10" and "2025-01" (or similar), here's how to proceed:

📘 Step 1: Filter the Data by Month

python

Copy

Edit

# Filter data for each month

oct\_2023\_df = df[df['year\_month'] == '2023-10']

jan\_2025\_df = df[df['year\_month'] == '2025-01']

🔍 Adjust the values above if your year\_month column format is different (e.g., October 2023 or Jan 2025).

📗 Step 2: Add Sentiment Label (Same as Before)

If you haven’t already labeled them, apply this to both:

python

Copy

Edit

def label\_sentiment(score):

if 0 <= score < 0.5:

return 'Negative'

elif 0.5 <= score < 1.5:

return 'Neutral'

elif 1.5 <= score <= 2:

return 'Positive'

else:

return 'Unknown'

oct\_2023\_df['sentiment\_label'] = oct\_2023\_df['avg\_sentiment'].apply(label\_sentiment)

jan\_2025\_df['sentiment\_label'] = jan\_2025\_df['avg\_sentiment'].apply(label\_sentiment)

📙 Step 3: Count Articles Per Sentiment (for each month)

python

Copy

Edit

oct\_counts = oct\_2023\_df['sentiment\_label'].value\_counts().reindex(['Positive', 'Neutral', 'Negative']).reset\_index()

oct\_counts.columns = ['Sentiment', 'Count']

jan\_counts = jan\_2025\_df['sentiment\_label'].value\_counts().reindex(['Positive', 'Neutral', 'Negative']).reset\_index()

jan\_counts.columns = ['Sentiment', 'Count']

📕 Step 4: Plot Simple Bar Charts (One for Each Month)

python

Copy

Edit

import plotly.express as px

# October 2023

fig\_oct = px.bar(oct\_counts, x='Sentiment', y='Count', title='Sentiment Distribution - October 2023')

fig\_oct.show()

# January 2025

fig\_jan = px.bar(jan\_counts, x='Sentiment', y='Count', title='Sentiment Distribution - January 2025')

fig\_jan.show()

✅ Output

This will give you two clean bar charts:

One for October 2023

One for January 2025

Each chart shows:

X-axis: Sentiment category (Positive, Neutral, Negative)

Y-axis: Number of articles in that category

Let me know if your year\_month values are in a different format (like "October 2023" instead of "2023-10"), and I can help adjust the filters accordingly.

You said:

I donot want to create two separate bar charts I want one with on x axis having sentiment lets say postive on which there will be two bars one as Jan-2024 bar and the other will be Oct 2023 color coded differently and on hovering you can see the counts of the articles

ChatGPT said:

Got it! You want one grouped bar chart (also called a clustered bar chart) where:

X-axis = sentiment categories (Positive, Neutral, Negative)

For each sentiment, you have two bars: one for October 2023 and one for January 2024 (we'll assume you meant January 2024 earlier)

Color = month (for visual distinction)

Hover = shows count of articles

✅ Step-by-Step for Grouped Bar Chart with Plotly Express

📘 Step 1: Apply Sentiment Labels

Make sure the main DataFrame (df) has the sentiment labels:

python

Copy

Edit

def label\_sentiment(score):

if 0 <= score < 0.5:

return 'Negative'

elif 0.5 <= score < 1.5:

return 'Neutral'

elif 1.5 <= score <= 2:

return 'Positive'

else:

return 'Unknown'

df['sentiment\_label'] = df['avg\_sentiment'].apply(label\_sentiment)

📗 Step 2: Filter Only the Two Months You Want

Adjust the month values according to your data format. Assuming year\_month is in "YYYY-MM" format:

python

Copy

Edit

filtered\_df = df[df['year\_month'].isin(['2023-10', '2024-01'])]

If your year\_month format is like "October 2023" and "January 2024", then use:

python

Copy

Edit

filtered\_df = df[df['year\_month'].isin(['October 2023', 'January 2024'])]

📙 Step 3: Group by sentiment\_label and year\_month to Get Counts

python

Copy

Edit

grouped = filtered\_df.groupby(['sentiment\_label', 'year\_month']).size().reset\_index(name='Count')

This gives a DataFrame like:

sentiment\_label year\_month Count

Positive 2023-10 20

Positive 2024-01 30

Neutral 2023-10 50

... ... ...

📕 Step 4: Plot with Plotly Express (Grouped Bars)

python

Copy

Edit

import plotly.express as px

fig = px.bar(

grouped,

x='sentiment\_label',

y='Count',

color='year\_month',

barmode='group',

title='Sentiment Comparison: October 2023 vs January 2024',

labels={'sentiment\_label': 'Sentiment', 'Count': 'Number of Articles', 'year\_month': 'Month'}

)

fig.show()

✅ Result:

X-axis: Sentiments (Positive, Neutral, Negative)

Two bars for each sentiment (one per month)

Color-coded by month (year\_month)

Interactive hover with exact counts

Let me know if:

Your month format is different

You want the bars sorted or ordered in a custom way

You’d like to save the chart as an HTML or image file

Happy plotting!

**ChatGPT Solution No.7:**

Okay so we created a absolute numbers bar plot, but now I want to explore creating a separate bar plot with relative numbers (percentage of the positive/neutral/negative articles per month), to make the two months more comparable given the difference in number of articles can you guide me how can I do that

ChatGPT said:

Absolutely! You're now moving from absolute counts to relative percentages, which is a great way to compare distributions across months—even if the total number of articles is different in each month.

We’ll follow a similar approach as before but normalize the data within each month.

✅ Goal: Grouped Bar Chart of Relative Percentages

X-axis: sentiment\_label (Positive, Neutral, Negative)

Two bars per sentiment: one per month (color-coded)

Y-axis: percentage (%) of articles in that sentiment within each month

🧠 Step-by-Step Guide

📘 Step 1: Group and Count as Before

python

Copy

Edit

grouped = df[df['year\_month'].isin(['2023-10', '2024-01'])] \

.groupby(['year\_month', 'sentiment\_label']) \

.size() \

.reset\_index(name='count')

📗 Step 2: Normalize Counts to Percentages (Per Month)

We’ll calculate the total articles per month, then divide each sentiment count by that total to get percentages.

python

Copy

Edit

# Calculate total articles per month

totals = grouped.groupby('year\_month')['count'].transform('sum')

# Add percentage column

grouped['percentage'] = (grouped['count'] / totals) \* 100

Now grouped looks like:

year\_month sentiment\_label count percentage

2023-10 Negative 30 30.0

2023-10 Neutral 50 50.0

2023-10 Positive 20 20.0

2024-01 Negative 10 10.0

2024-01 Neutral 20 20.0

2024-01 Positive 70 70.0

📙 Step 3: Plot Using Plotly Express

python

Copy

Edit

import plotly.express as px

fig = px.bar(

grouped,

x='sentiment\_label',

y='percentage',

color='year\_month',

barmode='group',

title='Relative Sentiment Distribution: October 2023 vs January 2024',

labels={'sentiment\_label': 'Sentiment', 'percentage': 'Percentage of Articles', 'year\_month': 'Month'},

text=grouped['percentage'].round(1) # optional: show % on bars

)

fig.update\_traces(textposition='outside')

fig.show()

🎯 What You Get:

A grouped bar chart where each sentiment has:

A bar for October 2023

A bar for January 2024

Y-axis is percentage, making cross-month comparisons meaningful even if article counts differ

Let me know if you’d like to:

Sort sentiments (e.g. always show Positive → Neutral → Negative)

Show percentages inside bars

Add hover tooltips with both count and percentage

Happy comparing!

**ChatGPT Solution No. 8:**

Hello ChatGPT so I have a csv file named avg\_sentiments\_results.csv which has four columns named filename, year\_month, title, avg\_sentiment. I have loaded this csv file as a panda dataframe in python and want to create a scatter plot using the plotly express libary. So in the csv file I have two months data anmed january 2024 and October 2023 first I want to filter the data based on months next I want to scatter plot that has months on x axis and avg\_sentiment scores on y axis which shows that avg sentiment from 0-0.5 is negative, 0.5-1.5 is neutral and 1.5-2 is positive and there will be points for indiviudal texts set up straight line on the month label on x-axis can you help me by acting as a tutor so that I can do the task I want to do step by step

ChatGPT said:

Of course! I'd be happy to guide you step by step like a tutor. We'll go through this in stages:

🔹 Step 1: Load Your CSV File

You mentioned you already loaded the CSV, but just to be sure, here’s how we do it:

python

Copy

Edit

import pandas as pd

df = pd.read\_csv('avg\_sentiments\_results.csv')

print(df.head())

✅ Action: Run this and confirm your columns (filename, year\_month, title, avg\_sentiment) are correctly loaded.

🔹 Step 2: Check Unique year\_month Values

Let’s check the values you have under the year\_month column:

python

Copy

Edit

print(df['year\_month'].unique())

✅ Action: This helps ensure that the values like 'January 2024' and 'October 2023' are written exactly like that (no typos or extra spaces).

Let me know what values you get before we move on to filtering!

You said:

yes I did this what next

ChatGPT said:

Great! Now that you've confirmed your year\_month values, let's move on.

🔹 Step 3: Filter the Data for January 2024 and October 2023

We want to keep only the rows for those two months:

python

Copy

Edit

filtered\_df = df[df['year\_month'].isin(['January 2024', 'October 2023'])]

print(filtered\_df.head())

✅ Action: Run this to confirm that your filtered dataframe only includes the rows for January 2024 and October 2023.

🔹 Step 4: Install and Import Plotly

If you haven’t installed Plotly yet:

bash

Copy

Edit

pip install plotly

Then import Plotly Express:

python

Copy

Edit

import plotly.express as px

Let me know once you’re done with filtering and importing so we can move to plotting!

You said:

okay what next

ChatGPT said:

Awesome! Now let’s move to the exciting part — creating the scatter plot. Here's what we want to achieve:

x-axis: year\_month (either "October 2023" or "January 2024")

y-axis: avg\_sentiment (individual points per text)

We’ll plot each text’s sentiment vertically aligned on its corresponding month

🔹 Step 5: Create a Scatter Plot with Plotly Express

python

Copy

Edit

fig = px.scatter(

filtered\_df,

x='year\_month',

y='avg\_sentiment',

hover\_data=['title'], # Shows the title when you hover

color='avg\_sentiment', # Optional: color points based on sentiment

color\_continuous\_scale='RdYlGn', # Red to Green (neg to pos)

)

# Add custom lines to show sentiment ranges

fig.add\_shape(type="line", x0=-0.5, x1=1.5, y0=0.5, y1=0.5,

line=dict(color="blue", dash="dash"), name="Neutral Lower Bound")

fig.add\_shape(type="line", x0=-0.5, x1=1.5, y0=1.5, y1=1.5,

line=dict(color="green", dash="dash"), name="Positive Lower Bound")

# Customize layout

fig.update\_layout(

title='Average Sentiment Scores by Month',

xaxis\_title='Month',

yaxis\_title='Average Sentiment',

yaxis=dict(range=[0, 2]), # Set fixed range from 0 to 2

)

fig.show()

🔍 What This Does

x='year\_month' puts each point under its month.

y='avg\_sentiment' puts it vertically based on the sentiment score.

The hover\_data=['title'] lets you see the text title when you hover.

Horizontal lines at 0.5 and 1.5 visually divide Negative, Neutral, and Positive ranges.

✅ Action: Run this code and check your scatter plot. Let me know if you’d like to:

Change colors

Label the sentiment zones ("Negative", etc.)

Separate plots for each month

Or add anything else!

You said:

# Classify sentiment into categories Positive, Negative, and Neutral

def classify\_sentiment(score):

if score < 0.5:

return "Negative"

elif score < 1.5:

return "Neutral"

else:

return "Positive"

df["sentiment\_category"] = df["avg\_sentiment"].apply(classify\_sentiment)

# Map months to human-readable labels

month\_map = {

"2023-10": "October 2023",

"2024-01": "January 2024"

}

df["month\_label"] = df["year\_month"].map(month\_map)

# Create a linear numeric x-axis for spacing

month\_to\_num = {

"October 2023": 0,

"January 2024": 1

}

df["month\_num"] = df["month\_label"].map(month\_to\_num)

# Plot scatter using numeric x-axis and label ticks manually

fig3 = px.scatter(

df,

x="month\_num",

y="avg\_sentiment",

color="sentiment\_category",

hover\_data={"title": True, "month\_num": False},

labels={"avg\_sentiment": "Average Sentiment"},

title="Sentiment of Articles: October 2023 (War) vs January 2024 (Ceasefire)",

opacity=0.5

)

# Fix x-axis to show labels instead of numbers

fig3.update\_layout(

xaxis=dict(

tickmode='array',

tickvals=[0, 1],

ticktext=["October 2023", "January 2024"],

title="Month"

),

yaxis=dict(title="Average Sentiment (0-0.5 = Negative, 0.5-1.5 = Neutral, 1.5-2 = Positive)", range=[0, 2]),

legend\_title="Sentiment Category"

)

fig3.show() so I took help from here and there and wrote this script do you think it does what I want to do because at some points the codes are complex and I donot understand what they are doing please breakdown them for me so that I can understand

ChatGPT said:

You're doing really well! Yes, your code does what you intended:

🔹 Filters the data by month

🔹 Classifies sentiment into categories

🔹 Places points vertically based on sentiment score

🔹 Aligns them horizontally using numeric x-values for spacing

🔹 Labels the x-axis with readable month names

🔹 Colors the points by sentiment category

Now let me break it down step by step, like a tutor, to make sure you understand each part clearly:

🧠 Part-by-Part Breakdown of Your Code

1. Classify Sentiment into Categories

python

Copy

Edit

def classify\_sentiment(score):

if score < 0.5:

return "Negative"

elif score < 1.5:

return "Neutral"

else:

return "Positive"

df["sentiment\_category"] = df["avg\_sentiment"].apply(classify\_sentiment)

🔹 This function checks the sentiment score:

< 0.5 → "Negative"

0.5 to <1.5 → "Neutral"

≥ 1.5 → "Positive"

🔹 Then .apply() is used to apply this function to each row in the column avg\_sentiment, creating a new column sentiment\_category.

✅ Result: Now each text is tagged as Positive / Neutral / Negative.

2. Map Machine-Readable Month Codes to Human-Readable Labels

python

Copy

Edit

month\_map = {

"2023-10": "October 2023",

"2024-01": "January 2024"

}

df["month\_label"] = df["year\_month"].map(month\_map)

🔹 You had months in format like "2023-10" or "2024-01".

🔹 This dictionary (month\_map) converts those codes to labels like "October 2023".

✅ Result: Now month\_label column has readable names for the x-axis.

3. Assign Numeric Values for Months

python

Copy

Edit

month\_to\_num = {

"October 2023": 0,

"January 2024": 1

}

df["month\_num"] = df["month\_label"].map(month\_to\_num)

🔹 Plotly can't space points on the x-axis if they're strings like "October 2023", so you map them to numbers:

October → 0

January → 1

✅ Result: Points now appear as vertical lines per month (like a strip).

4. Create the Scatter Plot

python

Copy

Edit

fig3 = px.scatter(

df,

x="month\_num",

y="avg\_sentiment",

color="sentiment\_category",

hover\_data={"title": True, "month\_num": False},

labels={"avg\_sentiment": "Average Sentiment"},

title="Sentiment of Articles: October 2023 (War) vs January 2024 (Ceasefire)",

opacity=0.5

)

🔹 x = month\_num: Puts points at 0 (October) or 1 (January)

🔹 y = avg\_sentiment: Puts them vertically by score

🔹 color = sentiment\_category: Colors dots as per Positive/Neutral/Negative

🔹 hover\_data: Shows the article title when you hover, but hides the numeric x-axis value

🔹 opacity = 0.5: Makes dots slightly transparent for overlapping clarity

5. Update the Axis Labels and Appearance

python

Copy

Edit

fig3.update\_layout(

xaxis=dict(

tickmode='array',

tickvals=[0, 1],

ticktext=["October 2023", "January 2024"],

title="Month"

),

yaxis=dict(title="Average Sentiment (0-0.5 = Negative, 0.5-1.5 = Neutral, 1.5-2 = Positive)", range=[0, 2]),

legend\_title="Sentiment Category"

)

🔹 You manually tell Plotly:

"At x=0, show 'October 2023'"

"At x=1, show 'January 2024'"

→ This makes the graph easier to read.

🔹 yaxis range is set from 0 to 2, and a descriptive title is added.

✅ Final Outcome

You now have:

Vertical scatter lines for each month

Color-coded sentiment

Interactive labels

Clear x and y axes

Would you like to:

Add horizontal bands or lines for the 0.5 and 1.5 thresholds?

Annotate each sentiment zone?

Export the plot as an image or HTML file?

Let me know and I’ll guide you further!

You said:

okay thank you so much completely understood the script

ChatGPT said:

You're very welcome — I'm really glad to hear that you fully understood it! 🙌

You're doing a great job combining data analysis with clear visual storytelling 👏

If you ever want to:

Animate sentiment over more months 📈

Combine with word clouds or bar charts 🧠

Add labels directly on the dots 🏷️

Or turn this into a report or dashboard 📊

Feel free to ask — I’m always here to help.

Good luck with your project! 💪✨

**ChatGPT Solution No. 9:**

Hello ChatGPT, I'm working on a Digital Humanities project where I am visualizing sentiment analysis data from a CSV file called "avg\_sentiment\_results.csv" located in the data/sentiment analysis folder. I want to create a bar chart that shows the relative percentage of positive, neutral, and negative articles for October 2023 and January 2024 to make comparisons easier despite the different number of articles published in each month. I’m thinking of grouping articles based on sentiment labels (Negative, Neutral, Positive) derived from the average sentiment score, calculating the percentage of each sentiment per month, and then plotting it using Plotly Express. Can you act as a Tutor and help me do this task step by step

ChatGPT said:

Absolutely, I'd be happy to guide you through this step-by-step as a tutor. Let's break down your goal into manageable parts and walk through each one. You're trying to:

Load your CSV data.

Filter it for the months of October 2023 and January 2024.

Assign sentiment labels based on the average sentiment score.

Calculate the relative percentage of each sentiment category per month.

Visualize the results with a bar chart using Plotly Express.

🧩 Step 1: Load Your CSV Data

You'll need pandas for handling the CSV data.

python

Copy

Edit

import pandas as pd

# Load the data

df = pd.read\_csv('data/sentiment analysis/avg\_sentiment\_results.csv')

# View the first few rows

print(df.head())

Let me know what the column names are, especially for:

Date (e.g. date or published\_at)

Sentiment score (e.g. avg\_sentiment\_score)

🧩 Step 2: Filter for October 2023 and January 2024

We’ll convert the date column to datetime and then filter.

python

Copy

Edit

# Convert the date column to datetime format

df['date'] = pd.to\_datetime(df['date'])

# Filter for October 2023 and January 2024

df\_filtered = df[(df['date'].dt.to\_period('M') == '2023-10') |

(df['date'].dt.to\_period('M') == '2024-01')]

# Check the results

print(df\_filtered['date'].dt.to\_period('M').value\_counts())

🧩 Step 3: Assign Sentiment Labels

Let’s define thresholds (you can change them depending on how your sentiment scores range, e.g., -1 to 1).

python

Copy

Edit

# Define sentiment label based on score

def label\_sentiment(score):

if score < -0.05:

return 'Negative'

elif score > 0.05:

return 'Positive'

else:

return 'Neutral'

df\_filtered['sentiment'] = df\_filtered['avg\_sentiment\_score'].apply(label\_sentiment)

df\_filtered['month'] = df\_filtered['date'].dt.strftime('%B %Y')

# Check the distribution

print(df\_filtered.groupby(['month', 'sentiment']).size())

🧩 Step 4: Calculate Percentages

We want to compute the relative frequency (percentage) of each sentiment per month.

python

Copy

Edit

# Count number of articles per sentiment per month

counts = df\_filtered.groupby(['month', 'sentiment']).size().reset\_index(name='count')

# Total articles per month

totals = counts.groupby('month')['count'].transform('sum')

# Calculate percentage

counts['percentage'] = (counts['count'] / totals) \* 100

print(counts)

🧩 Step 5: Visualize with Plotly Express

Now we can create a grouped bar chart.

python

Copy

Edit

import plotly.express as px

fig = px.bar(

counts,

x='month',

y='percentage',

color='sentiment',

barmode='group',

text='percentage',

labels={'percentage': 'Sentiment (%)'},

title='Sentiment Distribution for October 2023 and January 2024'

)

fig.update\_traces(texttemplate='%{text:.1f}%', textposition='outside')

fig.update\_layout(yaxis=dict(range=[0, 100]), uniformtext\_minsize=8, uniformtext\_mode='hide')

fig.show()

✅ At This Point

You should see a neat, comparative bar chart showing the percentage of Positive, Neutral, and Negative articles for each of the two months.

Would you like to go hands-on with your own data now? You can upload a sample of the CSV or tell me the column names, and I can help troubleshoot or tailor this further.

You said:

can you give me more simplified codes that I can understand I think you are doing a bit complex ocding

ChatGPT said:

Absolutely! Let's simplify each step so it's easy to follow — especially if you're still learning Python and data handling. We'll keep the code beginner-friendly with comments to explain what each line does.

✅ Step-by-Step Simplified Code

🧩 Step 1: Import and Load Your Data

python

Copy

Edit

import pandas as pd # pandas is used to work with data

# Load the CSV file

df = pd.read\_csv('data/sentiment analysis/avg\_sentiment\_results.csv')

# Show the first few rows to understand the data

print(df.head())

👉 Make sure you know:

What the date column is called (e.g. date)

What the sentiment score column is called (e.g. avg\_sentiment\_score)

🧩 Step 2: Filter for October 2023 and January 2024

python

Copy

Edit

# Convert the date column to proper date format

df['date'] = pd.to\_datetime(df['date'])

# Keep only articles from October 2023 and January 2024

df = df[(df['date'].dt.month == 10) & (df['date'].dt.year == 2023) |

(df['date'].dt.month == 1) & (df['date'].dt.year == 2024)]

# Create a simple month name column

df['month'] = df['date'].dt.strftime('%B %Y') # e.g., "October 2023"

🧩 Step 3: Add Sentiment Labels

We'll label each row as Positive, Neutral, or Negative based on its score.

python

Copy

Edit

# Simple function to decide sentiment

def get\_sentiment(score):

if score < -0.05:

return 'Negative'

elif score > 0.05:

return 'Positive'

else:

return 'Neutral'

# Apply the function to the column

df['sentiment'] = df['avg\_sentiment\_score'].apply(get\_sentiment)

🧩 Step 4: Count and Calculate Percentages

python

Copy

Edit

# Count how many articles fall into each sentiment per month

counts = df.groupby(['month', 'sentiment']).size().reset\_index(name='count')

# Total articles per month

total\_counts = counts.groupby('month')['count'].transform('sum')

# Add percentage column

counts['percentage'] = (counts['count'] / total\_counts) \* 100

🧩 Step 5: Plot the Bar Chart (using Plotly Express)

python

Copy

Edit

import plotly.express as px

# Create the bar chart

fig = px.bar(

counts,

x='month',

y='percentage',

color='sentiment',

barmode='group',

title='Sentiment Percentages for October 2023 and January 2024'

)

# Show the chart

fig.show()

💡 Summary of What We Did

Step Action

1 Loaded the CSV file using pandas

2 Filtered data to only keep October 2023 and January 2024

3 Labeled each article as Positive, Neutral, or Negative

4 Counted how many of each sentiment type per month and calculated percentages

5 Created a bar chart to compare sentiment between months

If you can share:

The actual column names in your CSV file

A few rows of sample data

…I can simplify it even more just for your file!

You said:

\# Convert the date column to proper date format df\['date'\] = pd.to\\_datetime(df\['date'\]) \# Keep only articles from October 2023 and January 2024 df = df\[(df\['date'\].dt.month == 10) & (df\['date'\].dt.year == 2023) | (df\['date'\].dt.month == 1) & (df\['date'\].dt.year == 2024)\] \# Create a simple month name column df\['month'\] = df\['date'\].dt.strftime('%B %Y') \# e.g., "October 2023"

i was doing this step 2 and on running the code it gave me KeyError: 'date' can you help me understand why it is happening

ChatGPT said:

Great question — you're getting a KeyError: 'date' because your CSV file doesn't have a column named exactly 'date'. Python is case-sensitive and column names must match exactly.

✅ Here's how to fix it step by step:

🛠 Step 1: Check Your Column Names

Run this code first to see what your column names really are:

python

Copy

Edit

print(df.columns)

This will output something like:

python

Copy

Edit

Index(['published\_at', 'avg\_sentiment\_score', ...], dtype='object')

Maybe your date column is named 'published\_at', 'timestamp', or something else.

🛠 Step 2: Use the Correct Column Name

Suppose your date column is 'published\_at', you should change the code from:

python

Copy

Edit

df['date'] = pd.to\_datetime(df['date'])

To:

python

Copy

Edit

df['published\_at'] = pd.to\_datetime(df['published\_at']) # use the actual name

df['month'] = df['published\_at'].dt.strftime('%B %Y')

df = df[(df['published\_at'].dt.month == 10) & (df['published\_at'].dt.year == 2023) |

(df['published\_at'].dt.month == 1) & (df['published\_at'].dt.year == 2024)]

🧠 Tip

If you're unsure about column names that have spaces or strange characters, you can also try:

python

Copy

Edit

print(list(df.columns))

This will help you copy-paste the column name exactly.

Would you like to paste the column names here so I can help you rewrite that line correctly?

You said:

okay so there is no date in the dataframe sorry for not clearing it out it is year\_month column where only two months data is present 2023-10 and 2024-1

ChatGPT said:

Perfect — thanks for clarifying! Since your dataset already has a year\_month column with values like "2023-10" and "2024-1", we don’t need to filter using dates. We'll use that column directly to filter and label the months.

Let’s update and simplify everything using year\_month. Here's your easy-to-follow version of the code:

✅ Simplified Code Based on year\_month

🧩 Step 1: Load the CSV

python

Copy

Edit

import pandas as pd

# Load the data

df = pd.read\_csv('data/sentiment analysis/avg\_sentiment\_results.csv')

# Check what the data looks like

print(df.head())

🧩 Step 2: Keep Only October 2023 and January 2024

python

Copy

Edit

# Filter to only keep the two months

df = df[df['year\_month'].isin(['2023-10', '2024-1'])]

# Create readable month labels

df['month'] = df['year\_month'].replace({

'2023-10': 'October 2023',

'2024-1': 'January 2024'

})

🧩 Step 3: Add Sentiment Labels

(Assuming your sentiment score is in a column named avg\_sentiment\_score)

python

Copy

Edit

# Define a function to label sentiments

def get\_sentiment(score):

if score < -0.05:

return 'Negative'

elif score > 0.05:

return 'Positive'

else:

return 'Neutral'

# Apply sentiment labels

df['sentiment'] = df['avg\_sentiment\_score'].apply(get\_sentiment)

🧩 Step 4: Count and Calculate Percentages

python

Copy

Edit

# Count number of articles per sentiment per month

counts = df.groupby(['month', 'sentiment']).size().reset\_index(name='count')

# Total articles per month

counts['total'] = counts.groupby('month')['count'].transform('sum')

# Calculate percentages

counts['percentage'] = (counts['count'] / counts['total']) \* 100

🧩 Step 5: Plot Using Plotly

python

Copy

Edit

import plotly.express as px

# Make the bar chart

fig = px.bar(

counts,

x='month',

y='percentage',

color='sentiment',

barmode='group',

title='Sentiment Comparison: October 2023 vs January 2024'

)

fig.show()

✅ What to Check Before Running

Your CSV must have: year\_month, avg\_sentiment\_score

Only values '2023-10' and '2024-1' should be present in year\_month

If you want, you can paste the output of df.head() and df.columns here and I’ll tailor it 100% to your actual data.

You said:

okay so I worked and modified the script according to my needs it runs without error but the graph it produces has only bar for october 2023 below I am attaching the my script can you identify the problem where I am going wrong # importing the required libaries for visualization

import pandas as pd

import plotly.express as px

# Loading our avg\_sentiment\_results.csv file as a panda dataframe

df = pd.read\_csv("../data/dataframes/sentiment-analysis/avg\_sentiment\_results.csv")

# Printing the head of the dataframe to explore columns and rows

print(df.head())

# Filter to only keep the two months

df = df[df['year\_month'].isin(['2023-10', '2024-1'])]

# Create readable month labels

df['month'] = df['year\_month'].replace({

'2023-10': 'October 2023',

'2024-1': 'January 2024'

})

# Define a function to label sentiments

def get\_sentiment(score):

if score < -0.05:

return 'Negative'

elif score > 0.05:

return 'Positive'

else:

return 'Neutral'

# Apply sentiment labels

df['sentiment'] = df['avg\_sentiment'].apply(get\_sentiment)

# Count number of articles per sentiment per month

counts = df.groupby(['month', 'sentiment']).size().reset\_index(name='count')

# Total articles per month

counts['total'] = counts.groupby('month')['count'].transform('sum')

# Calculate percentages

counts['percentage'] = (counts['count'] / counts['total']) \* 100

# Make the bar chart

fig = px.bar(

counts,

x='month',

y='percentage',

color='sentiment',

barmode='group',

title='Sentiment Comparison: October 2023 vs January 2024'

)

fig.show()

ChatGPT said:

Thanks for sharing your full script — it looks really solid and clean. Since the code runs without errors but the graph shows only October 2023, that means the issue is likely with your data rather than the code itself.

Let’s go step-by-step to troubleshoot this:

✅ 1. Confirm If January 2024 Data Is Present

Add this line right after you filter by year\_month:

python

Copy

Edit

print(df['month'].value\_counts())

This will print how many rows exist for each month. Example output:

yaml

Copy

Edit

October 2023 50

January 2024 0

If January 2024 shows 0, then the problem is your data doesn't include any rows where year\_month == '2024-1'.

✅ 2. Check the Exact Formatting of year\_month Column

Sometimes '2024-1' might be formatted as '2024-01' (with a zero). Let’s print unique values to be sure:

python

Copy

Edit

print(df['year\_month'].unique())

If it shows '2024-01' instead of '2024-1', then just update your filtering step like this:

python

Copy

Edit

df = df[df['year\_month'].isin(['2023-10', '2024-01'])]

df['month'] = df['year\_month'].replace({

'2023-10': 'October 2023',

'2024-01': 'January 2024'

})

✅ 3. If It's Still Not Working…

Print the full df after filtering to verify:

python

Copy

Edit

print(df[['year\_month', 'month', 'avg\_sentiment']])

Check if any rows have 'January 2024' in the month column.

✅ Summary: Likely Fix

Change this line:

python

Copy

Edit

df = df[df['year\_month'].isin(['2023-10', '2024-1'])]

To:

python

Copy

Edit

df = df[df['year\_month'].isin(['2023-10', '2024-01'])] # zero-padded month

And update the .replace() dictionary the same way.

Let me know what you see in df['year\_month'].unique() and I’ll confirm the fix.

You said:

it works perfectly now thank you so much for your help

ChatGPT said:

You're very welcome — I’m really glad it worked out! 🎉

You did a great job troubleshooting and adjusting the script to your needs — that’s exactly how you grow as a coder. If you ever want to:

Add more months,

Try a pie chart or animation,

Or connect this with other text analysis results,

I’d be happy to guide you through.

Good luck with your Digital Humanities project — sounds like a meaningful and data-rich exploration! 💡📊