**Conversation with ChatGPT**

**Conversation 1**

I am a student of digital humanities and i have to use regular expressions to find the different place names in Gaza written in articles. previously I wrote the code for a single column of names due to which alternate names which were spelled differently in the articles were missed but now I have to create a code to find alternate names from another column also, which I did. I wrote a regular expression also. Now I also have to make sure that the articles that i am searching for the place names should be of the current war and the filenames of the articles contain the date. how can I do it with a condition that would skip the articles before the start of the recent war in Gaza

**ChatGPT said:**

To filter out only the articles **from the current Gaza war (starting October 7, 2023)** and use **regular expressions** to match place names from multiple columns, you can follow these steps in Python:

**✅ Step-by-step Approach**

**1. Assumptions**

* You have a list or DataFrame of alternate place names in multiple columns (e.g., column1, column2).
* Your article files are named with dates, e.g., 2023-10-07\_article.txt, 2023-09-15\_article.txt.
* You are looping through article files in a directory and checking content.

**✅ Step 1: Load the Place Names**

If you're using a CSV file with multiple columns for place name variants:

python

CopyEdit

import pandas as pd

# Load CSV containing place names (example: 'place\_names.csv')

df = pd.read\_csv('place\_names.csv')

# Combine all non-null values from both columns into a single set

place\_names = set(df['column1'].dropna().tolist() + df['column2'].dropna().tolist())

# Create a regex pattern for all place names (escaping special chars and allowing word boundaries)

import re

place\_pattern = r'\b(?:' + '|'.join(map(re.escape, place\_names)) + r')\b'

**✅ Step 2: Filter Files by Date**

You want to process only files with dates **after or on 2023-10-07**.

python

CopyEdit

import os

from datetime import datetime

# Define start of the war

war\_start = datetime.strptime("2023-10-07", "%Y-%m-%d")

# Directory containing your article files

directory = "articles/"

for filename in os.listdir(directory):

try:

# Extract the date from filename assuming it starts with YYYY-MM-DD

file\_date\_str = filename.split("\_")[0] # e.g., "2023-10-08"

file\_date = datetime.strptime(file\_date\_str, "%Y-%m-%d")

# Skip articles before the war

if file\_date < war\_start:

continue

# Process the article

with open(os.path.join(directory, filename), 'r', encoding='utf-8') as f:

content = f.read()

# Search for place names using regex

matches = re.findall(place\_pattern, content, flags=re.IGNORECASE)

if matches:

print(f"{filename}: {matches}")

except ValueError:

# Skip files with invalid date format

continue

**✅ Explanation**

* re.escape() ensures that special characters in place names don't break the regex.
* \b ensures you match full words (e.g., “Rafah” not “giraffah”).
* datetime.strptime() is used to compare the article date with the war's start date.
* The try-except block handles unexpected filenames.

Would you like to extract and **save the results into a CSV file** or perform **frequency analysis** of place names too?

4o

**You said:**

explain the code

**ChatGPT said:**

Sure! Let's break down the code into understandable parts so you can see what each section is doing and why it’s important for your **Digital Humanities** project.

**🧩 Purpose of the Code**

To:

1. **Filter article files** to include only those written after **October 7, 2023** (start of current Gaza war).
2. **Extract place names** from article texts using **regular expressions**.
3. Use **two columns** of alternate place names from a CSV file.

**🔍 Step-by-step Explanation**

**🔹 1. Loading Place Names from CSV**

python

CopyEdit

import pandas as pd

df = pd.read\_csv('place\_names.csv')

* This reads a CSV file (place\_names.csv) that contains two columns of place name variants (e.g., 'column1' and 'column2').
* pandas is used because it makes handling structured data (like tables) very easy.

python

CopyEdit

place\_names = set(df['column1'].dropna().tolist() + df['column2'].dropna().tolist())

* dropna() removes any empty (NaN) cells.
* .tolist() turns the column into a list.
* + merges both columns into one list.
* set() removes duplicates.

So now, place\_names contains **all unique place names** from both columns.

**🔹 2. Create a Regex Pattern to Match These Names**

python

CopyEdit

import re

place\_pattern = r'\b(?:' + '|'.join(map(re.escape, place\_names)) + r')\b'

* re.escape() makes sure any special characters in place names (like . or -) are treated literally in regex.
* '|'.join(...) builds a regex pattern like: \b(?:Gaza|Rafah|Khan Yunis)\b
* \b ensures **word boundaries**, so "Rafah" matches as a word, not as part of "Giraffah".

**🔹 3. Filter Files by Date**

python

CopyEdit

import os

from datetime import datetime

war\_start = datetime.strptime("2023-10-07", "%Y-%m-%d")

* This sets the starting date of the current Gaza war using Python’s datetime module.

python

CopyEdit

for filename in os.listdir("articles/"):

* Loops through all files in the folder named "articles".

python

CopyEdit

file\_date\_str = filename.split("\_")[0]

file\_date = datetime.strptime(file\_date\_str, "%Y-%m-%d")

* Assumes filenames start with a date, e.g., 2023-10-08\_article.txt
* split("\_")[0] extracts the date part (2023-10-08).
* strptime(...) converts that string into a datetime object for comparison.

python

CopyEdit

if file\_date < war\_start:

continue

* Skips articles **before October 7, 2023**.

**🔹 4. Read File and Search with Regex**

python

CopyEdit

with open(os.path.join("articles", filename), 'r', encoding='utf-8') as f:

content = f.read()

* Opens the file and reads its content into a string.

python

CopyEdit

matches = re.findall(place\_pattern, content, flags=re.IGNORECASE)

* re.findall() searches for **all matches** of the place names in the article.
* IGNORECASE makes the search **case-insensitive** (so it finds "rafah" or "Rafah").

**🔹 5. Print Results If Matches Found**

python

CopyEdit

if matches:

print(f"{filename}: {matches}")

* Only prints filenames and place names if any were found in the content.

**🔹 6. Handling Errors (Optional but Important)**

python

CopyEdit

except ValueError:

continue

* Some files might not have a proper date at the start (like note.txt).
* This skips such files without crashing the program.

**✅ Summary**

| **Section** | **What it Does** |
| --- | --- |
| Load CSV | Collects place name variants |
| Regex | Matches names accurately |
| Filter by date | Only keeps relevant war-time articles |
| Read + Search | Finds place names in each article |
| Output | Shows articles where names are found |

Would you like help adapting this into a full script you can run or save results to a file like CSV or Excel?

4o

**You said:**

why do we have to import another library like the date time

**ChatGPT said:**

Great question!

We import the datetime library because Python needs a **reliable and accurate way to understand and compare dates**—and strings like "2023-10-07" are just text until you tell Python they represent actual calendar dates.

**📅 Why datetime is Necessary:**

Let’s break it down with an example:

**🔸 Without datetime (string comparison):**

python

CopyEdit

"2023-10-08" > "2023-2-15" # This gives True (but incorrectly)

This comparison is **unreliable** because it's comparing characters in strings, not real calendar dates.

**🔸 With datetime:**

python

CopyEdit

from datetime import datetime

d1 = datetime.strptime("2023-10-08", "%Y-%m-%d")

d2 = datetime.strptime("2023-02-15", "%Y-%m-%d")

d1 > d2 # Correctly gives True

Now Python understands:

* "October 8, 2023" is *after* "February 15, 2023"
* You can now do accurate date filtering: if file\_date >= war\_start:

**✅ What datetime.strptime() Does:**

It **parses a string** like "2023-10-08" and converts it into a datetime object.

python

CopyEdit

datetime.strptime("2023-10-08", "%Y-%m-%d")

* %Y = 4-digit year
* %m = 2-digit month
* %d = 2-digit day

Now file\_date becomes a real date object that Python can understand and compare properly.

**✅ Summary**

| **Why Use datetime?** | **What It Solves** |
| --- | --- |
| To compare dates | String comparison is inaccurate |
| To filter articles correctly | Ensures only war-related ones are included |
| To work with dates flexibly | You can sort, filter, subtract, format, etc. |

Would you like an example of how to get the **latest article** or sort articles by date using datetime too?

4o

**You said:**

isn't there another way to do that without importing the library just manually

**ChatGPT said:**

Yes, you **can compare dates manually as strings** ***if*** the format is strictly **YYYY-MM-DD**, but it's **less safe** and not recommended for more complex tasks.

Let me explain both **how it works** and **why it's risky**:

**✅ When Manual String Comparison Works**

If all your filenames **strictly start** with dates in this format:

text

CopyEdit

"2023-10-08\_article.txt"

"2023-09-15\_article.txt"

Then you can **safely compare them as strings**, because ISO format (YYYY-MM-DD) is designed to make string comparisons behave like date comparisons.

**✅ Example (no datetime library):**

python

CopyEdit

war\_start = "2023-10-07"

filename = "2023-10-08\_article.txt"

file\_date\_str = filename.split("\_")[0]

if file\_date\_str >= war\_start:

print("Article is from the current war.")

else:

print("Article is too old.")

This will work **because**:

* '2023-10-08' > '2023-10-07' is True
* The characters are compared **left to right**, just like real date order: year → month → day

**⚠️ Risks of Manual Comparison**

While it works in this specific format, it can break if:

* The format changes (e.g., 8-10-2023)
* There's no leading zero (2023-2-5)
* You want to do date math (e.g., how many days ago?)
* You mix formats (e.g., 2023/10/07 vs 2023-10-07)

**✅ Summary**

| **Method** | **Pros** | **Cons** |
| --- | --- | --- |
| datetime.strptime() | Safe, flexible, powerful | Requires import |
| Manual string comparison | Simple, fast for strict formats | Fragile, unsafe if format changes |

So yes — for **your current use case**, string comparison is acceptable **if your filenames are consistently YYYY-MM-DD\_...**.

Would you like me to modify the original code to use manual comparison instead of datetime?

4o

**You said:**

yes do that and the explain the whole code step by step

**ChatGPT said:**

Perfect — here's the modified code using **manual string comparison** instead of importing the datetime library. Then, I’ll explain it step by step.

**✅ Modified Code (No datetime import)**

python

CopyEdit

import os

import pandas as pd

import re

# 1. Load place names from two columns in the CSV

df = pd.read\_csv('place\_names.csv')

place\_names = set(df['column1'].dropna().tolist() + df['column2'].dropna().tolist())

# 2. Create a regex pattern for all place names

place\_pattern = r'\b(?:' + '|'.join(map(re.escape, place\_names)) + r')\b'

# 3. Define the war start date as a string (manual comparison)

war\_start = "2023-10-07"

# 4. Loop through files in the directory

for filename in os.listdir("articles/"):

try:

# 5. Get date part from filename

file\_date\_str = filename.split("\_")[0]

# 6. Compare date strings manually

if file\_date\_str < war\_start:

continue # Skip files before the war

# 7. Open and read the file

with open(os.path.join("articles", filename), 'r', encoding='utf-8') as f:

content = f.read()

# 8. Search for place names in the text using regex

matches = re.findall(place\_pattern, content, flags=re.IGNORECASE)

# 9. Print results if any place names are found

if matches:

print(f"{filename}: {matches}")

except Exception as e:

continue # Skip files that don't have the expected format

**🧩 Step-by-step Explanation**

**🔹 Step 1: Load CSV with Place Names**

python

CopyEdit

df = pd.read\_csv('place\_names.csv')

place\_names = set(df['column1'].dropna().tolist() + df['column2'].dropna().tolist())

* Reads the CSV into a DataFrame using pandas.
* Takes two columns (like column1 and column2), removes empty cells, converts them to lists, and merges them.
* set() removes any duplicate names so you're not checking for the same place multiple times.

**🔹 Step 2: Build a Regex Pattern**

python

CopyEdit

place\_pattern = r'\b(?:' + '|'.join(map(re.escape, place\_names)) + r')\b'

* Builds a **single regex pattern** that matches any of the place names using the | OR operator.
* \b ensures it matches **whole words only**.
* re.escape() safely handles special characters in names (like hyphens or apostrophes).

**🔹 Step 3: Set the War Start Date (as a String)**

python

CopyEdit

war\_start = "2023-10-07"

* This is the **cutoff date** for filtering articles.

**🔹 Step 4: Loop Through Article Files**

python

CopyEdit

for filename in os.listdir("articles/"):

* Goes through all files in the articles/ folder.

**🔹 Step 5: Extract Date from Filename**

python

CopyEdit

file\_date\_str = filename.split("\_")[0]

* Assumes filename format is like: 2023-10-08\_article.txt
* Takes the first part ("2023-10-08") before the underscore \_ to use as the article date.

**🔹 Step 6: Compare Dates Manually**

python

CopyEdit

if file\_date\_str < war\_start:

continue

* Since both file\_date\_str and war\_start are in the YYYY-MM-DD format, they can be safely compared as strings.
* If the file’s date is **before** the war started, it skips processing that article.

**🔹 Step 7: Read the Article File**

python

CopyEdit

with open(os.path.join("articles", filename), 'r', encoding='utf-8') as f:

content = f.read()

* Opens the article file and reads the full text into the variable content.

**🔹 Step 8: Search for Place Names**

python

CopyEdit

matches = re.findall(place\_pattern, content, flags=re.IGNORECASE)

* Looks for all the place names in the article text using the regex pattern you built.
* IGNORECASE makes the match case-insensitive.

**🔹 Step 9: Print Matching Results**

python

CopyEdit

if matches:

print(f"{filename}: {matches}")

* If any place names were found in the article, print the filename and the list of matched names.

**✅ Summary**

| **Part** | **Purpose** |
| --- | --- |
| CSV Reading | Load alternate spellings of place names |
| Regex | Efficient and flexible pattern matching |
| Manual Date Check | Filter only current-war articles using string comparison |
| File Looping | Read and check each article’s content |
| Output | See which articles mention which Gaza places |

Would you like to **save these results to a file** instead of printing them?

**Conversation 2**

when i run the overall code it just stays blank with this written on it = RESTART: C:\Users\Komal Ali\Downloads\FASDH25-portfolio2\regex\_script\_komal.py.py

**ChatGPT said:**

Thanks for pointing that out. A blank output after the script runs (especially with RESTART:) usually means:

**✅ Your script ran but:**

1. **No matching files met your conditions**, OR
2. **No place names were found** in the articles, OR
3. **There was an error**, but it was silently skipped due to try-except (and nothing was printed), OR
4. **Your print() statement never got triggered**

Let’s troubleshoot this step-by-step.

**✅ Step 1: Add a Debug Print to See Files Being Processed**

Right after this line:

python

CopyEdit

for filename in os.listdir("articles/"):

Add this:

python

CopyEdit

print("Checking file:", filename)

This will show **whether your loop is running** at all.

**✅ Step 2: Temporarily Remove the try-except**

Your current try block hides errors. Temporarily **remove or comment out** the try-except like this:

python

CopyEdit

# try:

file\_date\_str = filename.split("\_")[0]

if file\_date\_str < war\_start:

continue

with open(os.path.join("articles", filename), 'r', encoding='utf-8') as f:

content = f.read()

matches = re.findall(place\_pattern, content, flags=re.IGNORECASE)

if matches:

print(f"{filename}: {matches}")

# except Exception as e:

# continue

This way, if there's any **error in reading the file** or comparing dates, Python will **show it**, not hide it.

**✅ Step 3: Check These Common Issues**

**🔸 Is the articles/ folder in the same directory as the script?**

Make sure your folder is like:

yaml

CopyEdit

FASDH25-portfolio2/

│

├── regex\_script\_komal.py

├── place\_names.csv

└── articles/

├── 2023-10-08\_article1.txt

└── 2023-10-09\_article2.txt

If the folder is in a different location, Python won't find the files.

**🔸 Do the filenames start with a date like 2023-10-08\_?**

The line:

python

CopyEdit

file\_date\_str = filename.split("\_")[0]

only works if your filenames follow that structure. If your files are named like:

CopyEdit

gaza\_article1\_2023-10-08.txt

you'll need to extract the date differently (using regex or split from the end).

**🔸 Does your CSV file have the correct column names?**

Check that 'column1' and 'column2' exactly match your CSV headers. If they are like "Place\_Name\_1" and "Place\_Name\_2" instead, you must update this:

python

CopyEdit

place\_names = set(df['Place\_Name\_1'].dropna().tolist() + df['Place\_Name\_2'].dropna().tolist())

**✅ Summary Fixes**

Here’s a version that **prints what's happening** to help debug:

python

CopyEdit

import os

import pandas as pd

import re

df = pd.read\_csv('place\_names.csv')

place\_names = set(df['column1'].dropna().tolist() + df['column2'].dropna().tolist())

place\_pattern = r'\b(?:' + '|'.join(map(re.escape, place\_names)) + r')\b'

war\_start = "2023-10-07"

for filename in os.listdir("articles/"):

print("Checking file:", filename)

file\_date\_str = filename.split("\_")[0]

if file\_date\_str < war\_start:

print(f"Skipping {filename} — before war started")

continue

with open(os.path.join("articles", filename), 'r', encoding='utf-8') as f:

content = f.read()

matches = re.findall(place\_pattern, content, flags=re.IGNORECASE)

if matches:

print(f"{filename}: {matches}")

else:

print(f"{filename}: No matches found")

Try this version and let me know **what output you see** — I’ll help you interpret or fix it further.

4o

**You said:**

it isn't just this small part of the code i have another part where i have to print the mentions of the place name per month after the time part. i guess it is just taking time for processing it worked after a while

**ChatGPT said:**

Ah, that makes a lot of sense — thanks for the clarification!

If your full script includes:

* **Reading dozens or hundreds of files**
* **Using regex over full-text articles**
* **Aggregating results by month**

...then yes, it can take **noticeable time** to process, especially if your files are large or numerous. That explains why it seemed stuck — it was just busy working.

**✅ Tips to Speed Up or Monitor Processing**

If you'd like to **monitor or speed up** processing, here are a few useful additions:

**1. Progress Logging**

Add a line to show progress through the files:

python

CopyEdit

for i, filename in enumerate(os.listdir("articles/")):

print(f"[{i+1}] Processing {filename}...")

**2. Basic Timer**

Use this to measure how long your script takes:

python

CopyEdit

import time

start\_time = time.time()

# ... your main code ...

print(f"\nDone in {round(time.time() - start\_time, 2)} seconds.")

**3. Optional: Process Only a Few Files First (for testing)**

To test logic without waiting for all files:

python

CopyEdit

for i, filename in enumerate(os.listdir("articles/")):

if i >= 5: # Only process 5 files

break

# continue with your logic

**4. Avoid Duplicate Regex Work**

If you’re using the same regex over and over, consider **compiling it once**:

python

CopyEdit

compiled\_pattern = re.compile(place\_pattern, flags=re.IGNORECASE)

...

matches = compiled\_pattern.findall(content)

If you share the part of your script where you're printing **mentions per month**, I can also review it for performance tips or formatting improvements. Would you like to do that?