Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 43, in <module> alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()] AttributeError: 'list' object has no attribute 'split'

**ChatGPT said:**

The error message:

pgsql

CopyEdit

AttributeError: 'list' object has no attribute 'split'

is telling you that you're trying to call the .split() method on a **list**, but .split() is a **string method** — not a list method.

**The Problem Code:**

python

CopyEdit

alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()]

This line assumes alternatenames is a string — but it's actually a **list**. So calling .split(",") on it is invalid.

**✅ Fix:**

If alternatenames is already a list of names, then you **don’t need to split it**. Just process it directly:

python

CopyEdit

alternatenames\_list = [alt.strip().lower() for alt in alternatenames if alt.strip()]

**❓ If you're unsure what alternatenames looks like, you can add a quick debug line:**

python

CopyEdit

print(type(alternatenames), alternatenames)

Would you like help checking how alternatenames is being defined earlier in your script?

4o

**You said:**

for row in rows[1:]: columns = row.split("\t") asciiname = columns[0] main\_name = asciiname.lower() name = columns [4].lower() alternatenames = [5] alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()] main\_names[main\_name] = [main\_name, name] + alternatenames\_list patterns[name] = 0

**ChatGPT said:**

Thanks for sharing the code. The key issue is with this line:

python

CopyEdit

alternatenames = [5]

This line assigns a list containing the integer 5 to the variable alternatenames. This is not what you intended. You likely meant to **extract the value from the 6th column (columns[5])** of your row, which presumably contains the alternate names as a **comma-separated string**.

**✅ Here's the corrected version:**

python

CopyEdit

for row in rows[1:]:

columns = row.split("\t")

asciiname = columns[0]

main\_name = asciiname.lower()

name = columns[4].lower()

alternatenames = columns[5] # Fixed this line

alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()]

main\_names[main\_name] = [main\_name, name] + alternatenames\_list

patterns[name] = 0

**Optional Debug Tip:**

If you're unsure about how many columns each row has, you can insert a debug print:

python

CopyEdit

print(len(columns), columns)

Let me know if you want help validating your input format.

**You said:**

Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 44, in <module> main\_names[main\_name] = [main\_name, name] + alternatenames\_list NameError: name 'main\_names' is not defined. Did you mean: 'main\_name'?

**ChatGPT said:**

The new error:

pgsql

CopyEdit

NameError: name 'main\_names' is not defined. Did you mean: 'main\_name'?

means that Python encountered this line:

python

CopyEdit

main\_names[main\_name] = [main\_name, name] + alternatenames\_list

but **main\_names has not been defined** before this line.

**✅ Fix:**

Before the for loop, you need to define the dictionary main\_names (and likely patterns too, since you're using that as well):

python

CopyEdit

main\_names = {}

patterns = {}

for row in rows[1:]:

columns = row.split("\t")

asciiname = columns[0]

main\_name = asciiname.lower()

name = columns[4].lower()

alternatenames = columns[5]

alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()]

main\_names[main\_name] = [main\_name, name] + alternatenames\_list

patterns[name] = 0

Let me know if you'd like help validating or saving the output of main\_names and patterns after the loop.

**You said:**

folder = "articles" # define the patterns we want to search: # load the gazetteer from the tsv file: path = "../gazetteers/geonames\_gaza\_selection.tsv" with open(path, encoding="utf-8") as file: data = file.read() rows = data.split("\n") this is my latest code and this is my error Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 44, in <module> main\_names[main\_name] = [main\_name, name] + alternatenames\_list NameError: name 'main\_names' is not defined. Did you mean: 'main\_name'? for row in rows[1:]: columns = row.split("\t") asciiname = columns[0] main\_name = asciiname.lower() name = columns [4].lower() alternatenames = columns[5] alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()] main\_names[main\_name] = [main\_name, name] + alternatenames\_list

**ChatGPT said:**

Thanks for sharing the updated code and error.

**❗ Problem:**

You're using the dictionary main\_names here:

python

CopyEdit

main\_names[main\_name] = ...

but you **never created/initialized it** before this line.

**✅ Solution:**

You need to **define the dictionary main\_names** before the for loop begins, like this:

python

CopyEdit

main\_names = {} # <--- Add this line before your loop

If you're also using patterns[name] = 0 as in your earlier code, you'll need:

python

CopyEdit

patterns = {} # Define this as well

**✅ Fully Corrected Code:**

python

CopyEdit

folder = "articles"

# Load the gazetteer from the tsv file:

path = "../gazetteers/geonames\_gaza\_selection.tsv"

with open(path, encoding="utf-8") as file:

data = file.read()

rows = data.split("\n")

# Initialize dictionaries

main\_names = {}

patterns = {}

for row in rows[1:]: # Skip header row

if not row.strip():

continue # skip empty rows

columns = row.split("\t")

if len(columns) < 6:

continue # skip incomplete rows

asciiname = columns[0]

main\_name = asciiname.lower()

name = columns[4].lower()

alternatenames = columns[5]

alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()]

main\_names[main\_name] = [main\_name, name] + alternatenames\_list

patterns[name] = 0

This version ensures:

* You define the needed dictionaries.
* You skip empty or malformed rows to avoid index errors.

Would you like help using main\_names to do pattern matching in the articles folder?

**You said:**

for main\_name, other\_names in main\_names.items(): expression = r"\b("+"|".join(map(re.escape, other\_names)) + r")\b" matches = re.findall(expression, text) n\_matches = len(matches) # add the number of times it was found to the total frequency: if n\_matches>0: if main\_name not in mentions\_per\_month: mentions\_per\_month[main\_name] = {} mentions\_per\_months[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches)

**ChatGPT said:**

Thanks for sharing this next part of your script. The logic looks mostly correct, but there's **a small typo** that's likely causing your issue:

**❗ Error:**

python

CopyEdit

mentions\_per\_months[main\_name][month] = ...

Here, you're using mentions\_per\_months — but everywhere else you're using mentions\_per\_month.

**✅ Fix:**

Just correct the typo by changing mentions\_per\_months to mentions\_per\_month:

python

CopyEdit

for main\_name, other\_names in main\_names.items():

expression = r"\b(" + "|".join(map(re.escape, other\_names)) + r")\b"

matches = re.findall(expression, text)

n\_matches = len(matches)

if n\_matches > 0:

if main\_name not in mentions\_per\_month:

mentions\_per\_month[main\_name] = {}

mentions\_per\_month[main\_name][month] = mentions\_per\_month[main\_name].get(month, 0) + n\_matches

**🔁 Also make sure:**

* You've already imported re at the top of your script:

python

CopyEdit

import re

* You've defined mentions\_per\_month = {} **before** this loop.
* text and month are correctly defined in your loop or scope.

Would you like help looping through files in the articles folder and applying this logic per month?

**You said:**

import re import os import pandas as pd path = "../gazetteers/geonames\_gaza\_selection.tsv" with open(path, encoding="utf-8") as file: data = file.read() rows = data.split("\n") main\_names = {} for row in rows[1:]: columns = row.split("\t") asciiname = columns[0] main\_name = asciiname.lower() name = columns [4].lower() alternatenames = columns[5] alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()] main\_names[main\_name] = [main\_name, name] + alternatenames\_list mentions\_per\_month = {} for filename in os.listdir(folder): date = filename[:10] if date>= "2023-10-07": month = filename[:7] file\_path = os.path.join(folder, filename) with open(file\_path, "r", encoding="utf-8") as file: text = file.read() for main\_name, other\_names in main\_names.items(): expression = r"\b("+"|".join(map(re.escape, other\_names)) + r")\b" matches = re.findall(expression, text) n\_matches = len(matches) if n\_matches>0: if main\_name not in mentions\_per\_month: mentions\_per\_month[main\_name] = {} mentions\_per\_month[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches) for place, months in mentions\_per\_month.items(): print(f"{place}:") for months, n\_matches in months.items(): print(f"{months}:{n\_matches}")

**ChatGPT said:**

Your updated script looks very solid overall! 👏 You're correctly:

* Reading a gazetteer,
* Extracting names and alternatenames,
* Searching for them in article texts,
* Counting mentions per place and per month.

However, **a couple of improvements and corrections** will make your code more robust and bug-free:

**✅ 1. Define folder before using it**

Right now you use os.listdir(folder) before defining what folder is. Add this line before your file loop:

python

CopyEdit

folder = "articles"

**✅ 2. Prevent IndexError from malformed rows**

Some rows in your .tsv file might not have 6+ columns. You can safely skip them:

python

CopyEdit

for row in rows[1:]:

columns = row.split("\t")

if len(columns) < 6:

continue # skip incomplete rows

...

**✅ 3. Print formatting fix (repeated variable name months)**

Inside your final print loop, you're reusing the variable name months:

python

CopyEdit

for place, months in mentions\_per\_month.items():

print(f"{place}:")

for months, n\_matches in months.items(): # <- change `months` here

print(f"{months}:{n\_matches}")

Fix it like this to avoid confusion:

python

CopyEdit

for place, months in mentions\_per\_month.items():

print(f"{place}:")

for month, n\_matches in months.items():

print(f"{month}:{n\_matches}")

**✅ 4. Optional: Sort months and places for readability**

You can sort the output:

python

CopyEdit

for place in sorted(mentions\_per\_month.keys()):

print(f"{place}:")

for month in sorted(mentions\_per\_month[place].keys()):

print(f"{month}: {mentions\_per\_month[place][month]}")

Let me know if you want to export the results as a .csv or Excel file — it's easy to do with pandas.

**You said:**

Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 76, in <module> mentions\_per\_month[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches) NameError: name 'month' is not defined

**ChatGPT said:**

The error you're seeing — NameError: name 'month' is not defined — means that the variable month wasn't properly defined before it was used in this line:

python

CopyEdit

mentions\_per\_month[main\_name][month] = ...

**🔍 Why this happened**

You are extracting month like this:

python

CopyEdit

if date >= "2023-10-07":

month = filename[:7]

But if the condition if date >= "2023-10-07" is **not** true, then month is never defined — yet the code continues and tries to use it.

**✅ Fix**

Move all code that depends on month **inside** the if date >= "2023-10-07" block, like this:

python

CopyEdit

for filename in os.listdir(folder):

date = filename[:10]

if date >= "2023-10-07":

month = filename[:7]

file\_path = os.path.join(folder, filename)

with open(file\_path, "r", encoding="utf-8") as file:

text = file.read()

for main\_name, other\_names in main\_names.items():

expression = r"\b(" + "|".join(map(re.escape, other\_names)) + r")\b"

matches = re.findall(expression, text)

n\_matches = len(matches)

if n\_matches > 0:

if main\_name not in mentions\_per\_month:

mentions\_per\_month[main\_name] = {}

mentions\_per\_month[main\_name][month] = (

mentions\_per\_month[main\_name].get(month, 0) + n\_matches

)

Let me know if you'd like to add a progress counter or export the results!

**You said:**

Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 76, in <module> mentions\_per\_month[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches) NameError: name 'month' is not defined

**ChatGPT said:**

The error you're getting — NameError: name 'month' is not defined — still means that the variable month is being used outside of the block where it was defined.

Let's fix this **step by step** by making sure month is only used if it was defined. Here's the **corrected and complete code block** with month safely scoped:

**✅ FIXED CODE:**

python

CopyEdit

import re

import os

import pandas as pd

# Load gazetteer file

path = "../gazetteers/geonames\_gaza\_selection.tsv"

with open(path, encoding="utf-8") as file:

data = file.read()

rows = data.split("\n")

main\_names = {}

# Extract names from gazetteer

for row in rows[1:]:

if row.strip() == "":

continue

columns = row.split("\t")

if len(columns) < 6:

continue

asciiname = columns[0]

main\_name = asciiname.lower()

name = columns[4].lower()

alternatenames = columns[5]

alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()]

main\_names[main\_name] = [main\_name, name] + alternatenames\_list

# Folder with articles

folder = "articles"

mentions\_per\_month = {}

# Process articles

for filename in os.listdir(folder):

if not filename.endswith(".txt"):

continue

date = filename[:10]

if date >= "2023-10-07":

month = filename[:7]

file\_path = os.path.join(folder, filename)

with open(file\_path, "r", encoding="utf-8") as file:

text = file.read()

for main\_name, other\_names in main\_names.items():

expression = r"\b(" + "|".join(map(re.escape, other\_names)) + r")\b"

matches = re.findall(expression, text)

n\_matches = len(matches)

if n\_matches > 0:

if main\_name not in mentions\_per\_month:

mentions\_per\_month[main\_name] = {}

mentions\_per\_month[main\_name][month] = (

mentions\_per\_month[main\_name].get(month, 0) + n\_matches

)

# Print results

for place, months in mentions\_per\_month.items():

print(f"{place}:")

for m, n\_matches in months.items():

print(f" {m}: {n\_matches}")

**🔍 Key Fixes:**

* Ensured month is only defined **if** date >= "2023-10-07".
* All code that uses month is **inside that condition**.
* Skips empty rows and non-.txt files to avoid errors.
* Added safe checks for column length to avoid IndexError.

Would you like this exported to a CSV file for better analysis?

**You said:**

'''This is your starting script for today's Python class. This script contains the code we wrote last week to count the number of times each place in Gaza is mentioned in our corpus. Now, we want to store this count into a tsv file. I have written a function (write csv) to do this - but it has some mistakes in it. Please fix the mistakes and call the function to write the ''' import re import os import pandas as pd # fix this function! # define which folder to use: # NB: these are different articles than in the previous weeks folder = "../articles" # define the patterns we want to search: # load the gazetteer from the tsv file: path = "../gazetteers/geonames\_gaza\_selection.tsv" with open(path, encoding="utf-8") as file: data = file.read() rows = data.split("\n") main\_names = {} for row in rows[1:]: columns = row.split("\t") asciiname = columns[0] main\_name = asciiname.lower() name = columns [4].lower() alternatenames = columns[5] alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()] main\_names[main\_name] = [main\_name, name] + alternatenames\_list # build a dictionary of patterns from the place names in the first column: mentions\_per\_month = {} # count the number of times each pattern is found in the entire folder: for filename in os.listdir(folder): date = filename[:10] if date>= "2023-10-07": month = filename[:7] # build the file path: file\_path = os.path.join(folder, filename) #print(f"The path to the article is: {file\_path}") # load the article (text file) into Python: with open(file\_path, "r", encoding="utf-8") as file: text = file.read() # find all the occurences of the patterns in the text: for main\_name, other\_names in main\_names.items(): expression = r"\b("+"|".join(map(re.escape, other\_names)) + r")\b" matches = re.findall(expression, text) n\_matches = len(matches) # add the number of times it was found to the total frequency: if n\_matches>0: if main\_name not in mentions\_per\_month: mentions\_per\_month[main\_name] = {} mentions\_per\_month[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches) for place, months in mentions\_per\_month.items(): print(f"{place}:") for month, n\_matches in months.items(): print(f"{month}:{n\_matches}") # print the frequency of each pattern:

**ChatGPT said:**

Thanks for sharing your script! You're almost there. I'll do three things for you:

1. ✅ **Fix the broken part about month** (like before).
2. ✅ **Provide a corrected write\_csv() function** that saves the mentions\_per\_month dictionary into a proper TSV file.
3. ✅ **Show how to call the function**.

**✅ STEP 1: Fix the incorrect use of month**

You're setting month = filename[:7] **even for files before October 7, 2023**, which may lead to month being undefined when used. Let's fix that by **indenting correctly**.

**✅ STEP 2: Write the write\_csv() function**

We'll convert the nested dictionary to a DataFrame and write it to a .tsv.

**✅ FINAL SCRIPT (Fully Corrected & Working):**

python

CopyEdit

import re

import os

import pandas as pd

# Fix this function to save results to TSV

def write\_csv(data\_dict, output\_path):

# Flatten the dictionary into a list of rows

rows = []

for place, months in data\_dict.items():

for month, count in months.items():

rows.append({"Place": place, "Month": month, "Mentions": count})

# Convert to DataFrame

df = pd.DataFrame(rows)

# Optional: sort for easier readability

df.sort\_values(by=["Place", "Month"], inplace=True)

# Write to TSV

df.to\_csv(output\_path, sep="\t", index=False)

print(f"Data saved to {output\_path}")

# Define the folder

folder = "../articles"

# Load the gazetteer

path = "../gazetteers/geonames\_gaza\_selection.tsv"

with open(path, encoding="utf-8") as file:

data = file.read()

rows = data.split("\n")

main\_names = {}

for row in rows[1:]:

if row.strip() == "":

continue

columns = row.split("\t")

if len(columns) < 6:

continue

asciiname = columns[0]

main\_name = asciiname.lower()

name = columns[4].lower()

alternatenames = columns[5]

alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()]

main\_names[main\_name] = [main\_name, name] + alternatenames\_list

# Initialize dictionary

mentions\_per\_month = {}

# Go through articles

for filename in os.listdir(folder):

if not filename.endswith(".txt"):

continue

date = filename[:10]

if date >= "2023-10-07":

month = filename[:7]

file\_path = os.path.join(folder, filename)

with open(file\_path, "r", encoding="utf-8") as file:

text = file.read()

for main\_name, other\_names in main\_names.items():

expression = r"\b(" + "|".join(map(re.escape, other\_names)) + r")\b"

matches = re.findall(expression, text)

n\_matches = len(matches)

if n\_matches > 0:

if main\_name not in mentions\_per\_month:

mentions\_per\_month[main\_name] = {}

mentions\_per\_month[main\_name][month] = (

mentions\_per\_month[main\_name].get(month, 0) + n\_matches

)

# Print the results

for place, months in mentions\_per\_month.items():

print(f"{place}:")

for m, count in months.items():

print(f" {m}: {count}")

# Call the function to write TSV

write\_csv(mentions\_per\_month, "mentions\_output.tsv")

Let me know if you'd like the script to also **group months**, **handle missing values**, or create **a pivot table** for better visual analysis!

**You said:**

what did you change?

**ChatGPT said:**

Great question! Here's a clear breakdown of **what I changed and fixed** in your script:

**✅ 1. Fixed the month not defined error**

**Problem:**  
You defined month = filename[:7] **only inside an if condition**, so if date < "2023-10-07", the variable month was never defined—but your code still tried to use it later, causing the NameError.

**Fix:**  
I moved the rest of the code **inside the same if date >= "2023-10-07" block**, so month is only used where it's guaranteed to be defined.

python

CopyEdit

for filename in os.listdir(folder):

if not filename.endswith(".txt"):

continue

date = filename[:10]

if date >= "2023-10-07":

month = filename[:7]

...

**✅ 2. Wrote and fixed the write\_csv() function**

**Problem:**  
You had a placeholder write\_csv() function that wasn't defined or working.

**Fix:**  
I added a real implementation of write\_csv() that:

* Flattens your nested dictionary (mentions\_per\_month[place][month] = count)
* Creates a pandas DataFrame
* Writes it cleanly to a .tsv file using df.to\_csv(..., sep="\t")

python

CopyEdit

def write\_csv(data\_dict, output\_path):

rows = []

for place, months in data\_dict.items():

for month, count in months.items():

rows.append({"Place": place, "Month": month, "Mentions": count})

df = pd.DataFrame(rows)

df.sort\_values(by=["Place", "Month"], inplace=True)

df.to\_csv(output\_path, sep="\t", index=False)

**✅ 3. Added basic validation for the gazetteer loading**

**Problem:**  
Some rows in your TSV might be empty or malformed, which could crash your script.

**Fix:**  
I added:

python

CopyEdit

if row.strip() == "":

continue

if len(columns) < 6:

continue

to safely skip any bad rows.

**✅ 4. Added .txt file check**

**Problem:**  
If your article folder has non-text files (e.g. .DS\_Store on Mac), the script would fail.

**Fix:**  
I added this line to only process .txt files:

python

CopyEdit

if not filename.endswith(".txt"):

continue

**✅ 5. Optional but helpful: Sorted the final DataFrame**

Just for better readability in your .tsv output:

python

CopyEdit

df.sort\_values(by=["Place", "Month"], inplace=True)

Would you like help turning the final output into a **pivot table** or chart for analysis?

**You said:**

Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 78, in <module> mentions\_per\_month[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches) NameError: name 'month' is not defined

**ChatGPT said:**

The error you're seeing indicates that the month variable isn't defined at the time when you're trying to use it. Specifically, this happens because the month variable is being assigned **only inside the conditional block** if date >= "2023-10-07", and if the filename doesn't meet the condition, month doesn't exist.

**Fix:**

To solve this, I suggest two things:

1. **Move the code that processes the month variable inside the if date >= "2023-10-07" block** so that it's only assigned when the file meets the condition.
2. Ensure that the variable month is **defined before** you use it.

Here is the updated section of the code:

**Updated Code:**

python

CopyEdit

for filename in os.listdir(folder):

date = filename[:10]

if date >= "2023-10-07":

# Now it's safe to define 'month' inside the if block

month = filename[:7]

# build the file path:

file\_path = os.path.join(folder, filename)

# load the article (text file) into Python:

with open(file\_path, "r", encoding="utf-8") as file:

text = file.read()

# find all the occurrences of the patterns in the text:

for main\_name, other\_names in main\_names.items():

expression = r"\b(" + "|".join(map(re.escape, other\_names)) + r")\b"

matches = re.findall(expression, text)

n\_matches = len(matches)

# add the number of times it was found to the total frequency:

if n\_matches > 0:

if main\_name not in mentions\_per\_month:

mentions\_per\_month[main\_name] = {}

mentions\_per\_month[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches)

# Check the output

for place, months in mentions\_per\_month.items():

print(f"{place}:")

for month, n\_matches in months.items():

print(f"{month}: {n\_matches}")

**Key Changes:**

* The line month = filename[:7] is now inside the if date >= "2023-10-07" condition to ensure that month is only defined for files that meet the condition.
* The rest of the code operates with this month value in place.

This should fix the NameError you encountered because now the month will be defined before you use it in mentions\_per\_month[main\_name][month].

Let me know if this resolves the issue!

**You said:**

I have done all the things now I have to write a tsv file that is called "regex\_counts.tsv". Could you please give me the steps need to to do this task (only complete steps not solution). Below is my script and I have to do the last step '''This is your starting script for today's Python class. This script contains the code we wrote last week to count the number of times each place in Gaza is mentioned in our corpus. Now, we want to store this count into a tsv file. I have written a function (write csv) to do this - but it has some mistakes in it. Please fix the mistakes and call the function to write the ''' import re import os import pandas as pd # define which folder to use: folder = "../articles" # load the gazetteer from the tsv file: path = "../gazetteers/geonames\_gaza\_selection.tsv" with open(path, "r", encoding = "utf-8") as file: data = file.read() rows = data.strip().split("\n") main\_names = {} for row in rows[1:]: columns = row.split("\t") asciiname = columns[0].strip() main\_name = asciiname.lower() name = columns [4].strip().lower() alternatenames = columns[5] alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()] main\_names[main\_name] = [main\_name, name] + alternatenames\_list # build a dictionary of patterns from the place names in the first column: mentions\_per\_month = {} # count the number of times each pattern is found in the entire folder: for filename in os.listdir(folder): date = filename[:10] if date>= "2023-10-07": month = filename[:7] # build the file path: file\_path = os.path.join(folder, filename) #print(f"The path to the article is: {file\_path}") # load the article (text file) into Python: with open(file\_path, "r", encoding="utf-8") as file: text = file.read() # find all the occurences of the patterns in the text: for main\_name, other\_names in main\_names.items(): expression = r"\b("+"|".join(map(re.escape, other\_names)) + r")\b" matches = re.findall(expression, text, flags = re.IGNORECASE) n\_matches = len(matches) # add the number of times it was found to the total frequency: if n\_matches>0: if main\_name not in mentions\_per\_month: mentions\_per\_month[main\_name] = {} mentions\_per\_month[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches) for place, months in mentions\_per\_month.items(): print(f"{place}:") for month, n\_matches in months.items(): print(f"{month}:{n\_matches}") Need to do below one now # call the function to write your tsv file: columns = ["asciiname", "frequency"] tsv\_filename = "frequencies.tsv"

**ChatGPT said:**

Here are the **complete steps** you need to follow to write the regex\_counts.tsv file based on your script:

**✅ Steps to Write regex\_counts.tsv File**

1. **Create a list of rows** for the TSV:
   * For each main\_name in mentions\_per\_month
   * For each month under that main\_name
   * Store a row with three values: main\_name, month, and frequency
2. **Create a DataFrame** using pandas:
   * Use the list of rows from step 1
   * Define column names, e.g., ["asciiname", "month", "frequency"]
3. **Sort the DataFrame (optional but good practice)**:
   * Sort by asciiname and month to make the output organized
4. **Write the DataFrame to a .tsv file**:
   * Use df.to\_csv(...) with:
     + sep="\t" (for tab-separated)
     + index=False
     + Desired filename (e.g., "regex\_counts.tsv")
5. **Call this function from your main script** to execute the file-writing step

Let me know if you'd like me to check or fix your write\_csv function next.

**You said:**

I tried but it is showing the following error

**ChatGPT said:**

Please share the **exact error message** you're seeing — that will help me identify the issue quickly and guide you accordingly. Could you paste it here?

**You said:**

Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 939, in \_finalize\_columns\_and\_data columns = \_validate\_or\_indexify\_columns(contents, columns) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 986, in \_validate\_or\_indexify\_columns raise AssertionError( AssertionError: 2 columns passed, passed data had 3 columns The above exception was the direct cause of the following exception: Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 89, in <module> df = pd.DataFrame(rows, columns = columns) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\frame.py", line 851, in \_\_init\_\_ arrays, columns, index = nested\_data\_to\_arrays( File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 520, in nested\_data\_to\_arrays arrays, columns = to\_arrays(data, columns, dtype=dtype) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 845, in to\_arrays content, columns = \_finalize\_columns\_and\_data(arr, columns, dtype) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 942, in \_finalize\_columns\_and\_data raise ValueError(err) from err ValueError: 2 columns passed, passed data had 3 columns

**ChatGPT said:**

This error means that your rows variable contains lists with **3 values**, but your columns list only defines **2 column names** — so Pandas doesn't know how to label the third column.

Here’s what you need to do:

**✅ Step-by-step Fix:**

1. **Inspect your rows data**  
   Check what you’re appending to rows. For example:

python

CopyEdit

rows.append([main\_name, month, n\_matches])

1. **Update the columns list to match**  
   If each row has 3 elements (like above), then your columns must have 3 labels. So update:

python

CopyEdit

columns = ["asciiname", "month", "frequency"]

1. **Create the DataFrame**

python

CopyEdit

df = pd.DataFrame(rows, columns=columns)

1. **Write to the TSV**

python

CopyEdit

df.to\_csv("regex\_counts.tsv", sep="\t", index=False)

Let me know if you'd like help writing this final part into your script.

**You said:**

I tried to follow the above steps but got following error Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 85, in <module> for main\_name, months in mentions\_per\_month: ValueError: too many values to unpack (expected 2)

**ChatGPT said:**

That error means you're trying to unpack something that isn’t a list of key-value pairs. Specifically:

python

CopyEdit

for main\_name, months in mentions\_per\_month:

This is incorrect because mentions\_per\_month is a **dictionary**, so to iterate over both the keys and values, you need to use .items().

**✅ Fix it like this:**

python

CopyEdit

for main\_name, months in mentions\_per\_month.items():

This will correctly unpack each main\_name and its corresponding months dictionary.

**Then, if you're building the rows list to save in the TSV, use:**

python

CopyEdit

rows = []

for main\_name, months in mentions\_per\_month.items():

for month, n\_matches in months.items():

rows.append([main\_name, month, n\_matches])

**And write it to TSV:**

python

CopyEdit

import pandas as pd

columns = ["asciiname", "month", "frequency"]

df = pd.DataFrame(rows, columns=columns)

df.to\_csv("regex\_counts.tsv", sep="\t", index=False)

Let me know if you'd like me to check your full script with this final part added.

**You said:**

I believe there is a role of nested loops here if I am not wrong

**ChatGPT said:**

Yes, you're absolutely right — **nested loops are necessary** here because your data structure is a **nested dictionary**:

python

CopyEdit

mentions\_per\_month = {

"gaza city": {

"2023-10": 15,

"2023-11": 8

},

"rafah": {

"2023-10": 5

},

...

}

So to write this as rows in a TSV (one row per place per month), you need:

**✅ A nested loop:**

1. Outer loop → iterates over each place (main\_name)
2. Inner loop → iterates over each month and count for that place

**Example:**

python

CopyEdit

rows = []

for main\_name, months in mentions\_per\_month.items(): # Outer loop

for month, n\_matches in months.items(): # Inner loop

rows.append([main\_name, month, n\_matches]) # Build each row

Then you convert the list rows into a DataFrame and save it to TSV.

Would you like me to integrate this directly into your full script for clarity?

**You said:**

Got below error a lot of struggle Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 939, in \_finalize\_columns\_and\_data columns = \_validate\_or\_indexify\_columns(contents, columns) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 986, in \_validate\_or\_indexify\_columns raise AssertionError( AssertionError: 2 columns passed, passed data had 3 columns The above exception was the direct cause of the following exception: Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 89, in <module> df = pd.DataFrame(rows, columns = columns) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\frame.py", line 851, in \_\_init\_\_ arrays, columns, index = nested\_data\_to\_arrays( File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 520, in nested\_data\_to\_arrays arrays, columns = to\_arrays(data, columns, dtype=dtype) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 845, in to\_arrays content, columns = \_finalize\_columns\_and\_data(arr, columns, dtype) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\construction.py", line 942, in \_finalize\_columns\_and\_data raise ValueError(err) from err ValueError: 2 columns passed, passed data had 3 columns

**ChatGPT said:**

Thanks for sharing the full error — it's clear now:

**❗ Problem:**

You're passing **3 data values per row** ([main\_name, month, n\_matches]), but telling pandas.DataFrame() to expect only **2 column names**.

**✅ Solution:**

You need to provide **3 column names** to match your row structure.

**✅ Fix:**

Update this line:

python

CopyEdit

columns = ['location', 'month', 'mention\_count']

Then create the DataFrame like this:

python

CopyEdit

df = pd.DataFrame(rows, columns=columns)

**✅ Full Working Snippet:**

python

CopyEdit

rows = []

for main\_name, months in mentions\_per\_month.items():

for month, n\_matches in months.items():

rows.append([main\_name, month, n\_matches])

columns = ['location', 'month', 'mention\_count']

df = pd.DataFrame(rows, columns=columns)

df.to\_csv("mentions\_per\_month.tsv", sep='\t', index=False)

Let me know if you'd like help visualizing or analyzing this data afterward!

**You said:**

Now I got following error Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 90, in <module> df.sort\_values(by =["asciiname", "month"], inplace = True) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\frame.py", line 7172, in sort\_values keys = [self.\_get\_label\_or\_level\_values(x, axis=axis) for x in by] File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\generic.py", line 1911, in \_get\_label\_or\_level\_values raise KeyError(key) KeyError: 'month'

**ChatGPT said:**

Thanks for the update — this new error means that **pandas can’t find a column called 'month'** in your DataFrame.

**✅ Most likely cause:**

You **renamed your columns to ['location', 'month', 'mention\_count']**, but are trying to sort by ['asciiname', 'month']. 'asciiname' doesn't exist anymore.

**✅ Solution:**

Update your sort line to match the actual column names in your DataFrame.

Replace:

python

CopyEdit

df.sort\_values(by=["asciiname", "month"], inplace=True)

With:

python

CopyEdit

df.sort\_values(by=["location", "month"], inplace=True)

**✅ Optional Check:**

To verify your column names before sorting, you can print them:

python

CopyEdit

print(df.columns)

This will help you avoid typos or mismatched column names.

Let me know if you'd like to group, plot, or analyze the data next!

**You said:**

'''This is your starting script for today's Python class. This script contains the code we wrote last week to count the number of times each place in Gaza is mentioned in our corpus. Now, we want to store this count into a tsv file. I have written a function (write csv) to do this - but it has some mistakes in it. Please fix the mistakes and call the function to write the ''' import re import os import pandas as pd # define which folder to use: folder = "../articles" # load the gazetteer from the tsv file: path = "../gazetteers/geonames\_gaza\_selection.tsv" with open(path, "r", encoding = "utf-8") as file: data = file.read() rows = data.strip().split("\n") main\_names = {} for row in rows[1:]: columns = row.split("\t") asciiname = columns[0].strip() main\_name = asciiname.lower() name = columns [4].strip().lower() alternatenames = columns[5] alternatenames\_list = [alt.strip().lower() for alt in alternatenames.split(",") if alt.strip()] main\_names[main\_name] = [main\_name, name] + alternatenames\_list # build a dictionary of patterns from the place names in the first column: mentions\_per\_month = {} # count the number of times each pattern is found in the entire folder: for filename in os.listdir(folder): date = filename[:10] if date>= "2023-10-07": month = filename[:7] # build the file path: file\_path = os.path.join(folder, filename) #print(f"The path to the article is: {file\_path}") # load the article (text file) into Python: with open(file\_path, "r", encoding="utf-8") as file: text = file.read() # find all the occurences of the patterns in the text: for main\_name, other\_names in main\_names.items(): expression = r"\b("+"|".join(map(re.escape, other\_names)) + r")\b" matches = re.findall(expression, text, flags = re.IGNORECASE) n\_matches = len(matches) # add the number of times it was found to the total frequency: if n\_matches>0: if main\_name not in mentions\_per\_month: mentions\_per\_month[main\_name] = {} mentions\_per\_month[main\_name][month] = (mentions\_per\_month[main\_name].get(month, 0) + n\_matches) for place, months in mentions\_per\_month.items(): print(f"{place}:") for month, n\_matches in months.items(): print(f"{month}:{n\_matches}") # call the function to write your tsv file: columns = ["asciiname", "months","frequency"] tsv\_filename = "regex\_counts.tsv" import pandas as pd rows = [] for main\_name, months in mentions\_per\_month.items(): for month, n\_matches in months.items(): rows.append([main\_name, month, n\_matches]) df = pd.DataFrame(rows, columns = columns) df.sort\_values(by =["asciiname", "month"], inplace = True) df.to\_csv(tsv\_filename, sep="\t", index = False) above is the complete script that I ran but got below error Traceback (most recent call last): File "C:\Users\DANISH LAPTOP\Downloads\FASDH25-portfolio2\scripts\regex\_script\_mubashir\_alizain\_iqra.py.py", line 90, in <module> df.sort\_values(by =["location", "month"], inplace = True) File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\frame.py", line 7172, in sort\_values keys = [self.\_get\_label\_or\_level\_values(x, axis=axis) for x in by] File "C:\Users\DANISH LAPTOP\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\generic.py", line 1911, in \_get\_label\_or\_level\_values raise KeyError(key) KeyError: 'location'

**ChatGPT said:**

Thank you for sharing your full script and the error trace — the issue is now very clear.

**🔍 Problem:**

You are trying to sort the DataFrame with:

python

CopyEdit

df.sort\_values(by=["location", "month"], inplace=True)

But your DataFrame df was created using:

python

CopyEdit

columns = ["asciiname", "months", "frequency"]

**✅ So the actual column names are:**

* "asciiname"
* "months" ← note the **s** at the end!
* "frequency"

**❗There are two problems:**

1. You are sorting by "location" — but that column **does not exist** (you meant "asciiname").
2. You are sorting by "month" — but your column is called "months" with an **extra 's'**.

**✅ Solution:**

Change your sort line to this:

python

CopyEdit

df.sort\_values(by=["asciiname", "months"], inplace=True)

**✅ Optional improvement:**

To make your code more consistent, it's better to rename "months" to "month" for clarity (since each row refers to one month only).

So, update this line:

python

CopyEdit

columns = ["asciiname", "months", "frequency"]

To:

python

CopyEdit

columns = ["asciiname", "month", "frequency"]

Then, your sort line becomes:

python

CopyEdit

df.sort\_values(by=["asciiname", "month"], inplace=True)

**✅ Final fix:**

python

CopyEdit

columns = ["asciiname", "month", "frequency"]

df = pd.DataFrame(rows, columns=columns)

df.sort\_values(by=["asciiname", "month"], inplace=True)

df.to\_csv(tsv\_filename, sep="\t", index=False)

Let me know if you'd like help plotting or analyzing the data next!

**You said:**

please give me steps to write a readme file

**ChatGPT said:**

Writing a clear and effective **README** file is essential for documenting your project and making it understandable and reusable for others (and your future self). Below are step-by-step instructions for writing a good README.md file — especially tailored for your **Gaza place name mentions project** using regex and NER.

**✅ Step-by-Step Guide to Write a README.md**

**1. Title**

markdown

CopyEdit

# Gaza Place Name Mentions Analysis

Give your project a clear, descriptive title.

**2. Project Overview / Description**

markdown

CopyEdit

## Overview

This project identifies and maps mentions of places in Gaza across a collection of news articles. It uses two techniques:

- Regular expressions with a gazetteer

- Named Entity Recognition (NER)

The goal is to analyze how frequently specific locations in Gaza are mentioned over time, especially after October 7, 2023.

**3. Folder Structure**

markdown

CopyEdit

## Folder Structure

Explain what each folder or file does:

markdown

CopyEdit

- `scripts/`: Contains the Python scripts for regex and NER processing.

- `articles/`: Folder with the text files of news articles.

- `gazetteers/`: Contains the Gaza place name gazetteer used for regex matching.

- `outputs/`: Contains TSV and CSV files with results (e.g., regex\_counts.tsv).

**4. How It Works / Steps**

markdown

CopyEdit

## How It Works

### Regex Method:

1. Load Gaza place names and their alternate names from a TSV gazetteer.

2. Loop through each article published after October 7, 2023.

3. Use regular expressions to match mentions of place names.

4. Count and aggregate mentions by month.

5. Output the results into a TSV file (`regex\_counts.tsv`).

### Named Entity Recognition (NER) Method:

1. Load the articles using `spacy`.

2. Extract all entities labeled as GPE (Geo-Political Entities).

3. Filter to only include those found in the Gaza gazetteer.

4. Aggregate counts by month.

5. Output the results into a CSV/TSV file (`ner\_counts.tsv`).

**5. Requirements / Dependencies**

markdown

CopyEdit

## Requirements

- Python 3.10 or later

- pandas

- spacy

Install dependencies:

bash

CopyEdit

pip install pandas spacy

python -m spacy download en\_core\_web\_sm

**6. Usage Instructions**

markdown

CopyEdit

## How to Run

1. Place all article `.txt` files inside the `articles/` folder.

2. Make sure your gazetteer is in `gazetteers/geonames\_gaza\_selection.tsv`.

3. Run the regex script:

```bash

python scripts/regex\_script.py

1. Check the output in regex\_counts.tsv.

yaml

CopyEdit

---

### 7. \*\*Output Samples / Visualizations (optional)\*\*

```markdown

## Output Sample

A preview of `regex\_counts.tsv`:

python-repl

CopyEdit

asciiname month frequency

gaza 2023-10 27

rafah 2023-11 15

...

**📊 Optional: Add graphs if available:**

You can include a bar chart or heatmap of place mentions over time using matplotlib or seaborn.

**8. Comparison of Methods (Optional but Ideal)**

markdown

CopyEdit

## Comparison of Regex vs NER

| Method | Strengths | Limitations |

|--------|-----------|-------------|

| Regex with Gazetteer | High precision for known place names | Misses entities not in gazetteer |

| NER (Spacy) | Can catch unexpected mentions | May have false positives or miss regional names |

**9. Author / Credit**

markdown

CopyEdit

## Author

This project was developed by [Your Names Here] as part of the Digital Humanities course at FAS-DH25.

**10. License (Optional)**

If you're publishing your code, consider adding a license.

markdown

CopyEdit

## License

This project is licensed under the MIT License.

**✅ Let me know:**

Would you like me to generate a full README.md for your project based on this structure?

4o